SAMS

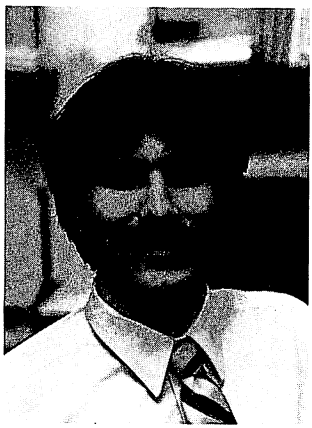
22395

TI-99/4A™ Trivia Data Base

James F. Hunter
and Gregory L. Guntle

TI-99/4A™ Trivia Data Base

James F. Hunter is currently Director of Publishing for Howard W. Sams & Co., Inc. (ITT). A graduate of the University of California (Riverside), Jim is a veteran of seven years experience in the personal computer field. In his spare time, he plays all board games with an enthusiasm and facility that sometimes astonish his opponents.



Gregory L. Guntle, a computer support specialist at Sams, is a 1983 graduate of Indiana University where he majored in computer science. He has worked actively with micros for the past six years. He also enjoys spending time in outdoors activities with his wife and family.

TI-99/4A™ Trivia Data Base

by

James F. Hunter
and
Gregory L. Guntle

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
FIRST PRINTING — 1984

International Standard Book Number: 0-672-22395-3
Library of Congress Catalog Card Number: 84-51277

Edited by *Douglas P. DeBrabant*

Printed in the United States of America.

Preface

All computers, including micros, are designed to emulate processes of the human mind. It is, after all, we who have defined the tasks assigned to computers, with the goal of freeing ourselves from tedious and repetitious tasks which can be done more quickly and efficiently by an electronic device.

It is not unexpected that, after working with computers for a while, we begin to regard them as intelligent living entities. While not accurate technically, such an attitude can be of use in discussing what the programs in this book are designed to do.

We need not be intimidated by the speed with which a computer can do repetitious tasks and calculations. Remember that, while doing those calculations, the computer need not be concerned with satisfying superiors on the job, raising children, paying bills, or trying to achieve goals set for itself, by itself. In short, a computer is not distracted by the state of being human.

For its part, the computer does not enjoy some of the very positive attributes of a human. It can not think, or feel, or play. We can. Playing a data retrieval game against a computer would be no fun. The computer would always win. It is our own lack of perfection at manipulating and retrieving data that has accounted for the tremendous success of games like *Trivial Pursuit*. As we shall learn, the process of information storage and retrieval in a computer is exact and describable. Not so with us humans.

How many times have you heard someone say, "That reminds me of a story." Why? What is the mechanism which links one thought or event to another in the human mind? I don't have the answer, but I do believe that the lack of exact precision in describing those links can be a source of entertainment for people.

We are by nature curious. In the exercise of that curiosity we amass tremendous quantities of information, some of which might not be of immediate or even long-term use. In an effort to justify the acquisition and retention of such bits of knowledge,

we at once name them trivia, and proudly proclaim ourselves true fountains of useless information. In simpler terms, we try simply to express our belief that knowing things for their own sake is rewarding, and fun.

The purposes of this book, and the included programs, are simple. First, to learn about the concept of a data base program on a computer, how it is developed, and how it works. Second, and perhaps more important, to take advantage of the given data base by using it as a pool from which to draw questions for the trivia random inquiry game program. Third, it's designed for you just to have fun. It is our hope that you will find both educational benefit and enjoyment in this book.

JAMES F. HUNTER

A NOTE TO THE READER

The programs in this book were not written as applications software but as educational examples of what your personal computer can do. All of the programs have been tested and work on the machine configuration for which they were designed. The programs are unprotected. This means that you can modify them to better understand how they work or to fit a different machine configuration.

What Is a Combo Pack?

A Combo Pack, like this package, is a step beyond your average technical book. While most books give you programming examples through printed listings (which we do here), Combo Packs provide the book and the listings recorded on magnetic media, either disk, cassette tape, or both.

Every effort has been made to be clear, concise, and informative about how these programs and routines work. If you experience any difficulty with the software operations, the solution can be found in the book or in your computer manuals.

We are rather proud of the time and effort that went into preparing the Combo Pack. If you have purchased the Combo Pack and have enjoyed using it, let us know your thoughts. Your comments will be valuable in preparing future Combo Packs.

LOADING INSTRUCTIONS

The cassette accompanying this Combo Pack contains the program listings printed in the book.

To load a cassette file from this tape, perform the following steps:

1. Put the cassette tape into the cassette recorder.
2. Position the tape at the beginning of the program you want to load.
3. Type **OLD CSI**
   Press <ENTER>

This will cause the next program on the tape to load into the computer's memory. When the program is loaded, it is ready to be used as described in the book.

After you have loaded the programs from cassette, you can, if you want, save them to disk. For information on how to do that, consult your computer manual.

The following list shows the tape counter positions for the contents of the cassette tape. These numbers are approximate and may vary from recorder to recorder. They should, however, assist you in locating the programs you are searching for.

Tape Directory

Program Name	Counter Location
Cassette Version – Data Base	Ø
Cassette Version – Game	4Ø
Diskette Version – Data Base	75
Diskette Version – Game	124
Sample File	178

Contents

Chapter 1
Introduction

In the 1940's, my father tuned pipe organs as a sideline and he later worked on the first electronic organs. In the process of repairing and tuning those organs, he kept running into the fact that the twelfth root of two is a very significant number in understanding the tempered musical scale. And he needed to understand that scale thoroughly to do a good job of tuning. Because he worked principally as a motion picture projectionist at the time, he had ample opportunity to set about calculating the twelfth root of two by hand.

The process was simple, if time consuming. He would pick a number between one and two, multiply it by itself twelve times, and see how close the result was to 2.00000. If the result of the multiplications (all done by hand) was greater than 2.00000, then he reduced the trial number. If it was less than 2.00000, then he increased the trial number. Over a period of *years*, he was able to calculate the twelfth root of two to 7 decimal places.

In 1975, I purchased a Hewlett Packard calculator. With a few keystrokes I found the log of 2, divided it by 12, and took the antilog. In a matter of *seconds* I'd obtained the same answer that it had taken my father years to arrive at. That speed in calculation is really what computers are all about. From the earliest modern computer, which was used by the U. S. Army to calculate mortar trajectories, to today's mainframe, mini, and microcomputers, the object has always been the same — to relegate repetitious and time consuming tasks to electromechanical (and now silicon technology) devices.

Repeated calculations to obtain a mathematical answer is popularly called *number crunching*. It was not long, however, after the advent of computers until other kinds of activities, which had previously been done by hand, were being done by computer. One very obvious application is financial accounting, with its ledgers, balance sheets, T accounts, and profit and loss

statements. One has but to remember the frustration of trying to balance a checkbook to understand the relief felt by accountants with the introduction of computerized bookkeeping.

As computers have become smaller, more affordable, and more powerful, other applications have been defined and implemented. Specifically, the applications which are of interest here are those currently being used on microcomputers, such as the TI-99/4A. It is helpful to understand general groupings of those applications, so as to put the programs contained in this book and the accompanying cassette in perspective. There are five main categories of microcomputer software, and each has several subcategories:

- Accounting
  - General Ledger
  - Accounts Payable
  - Accounts Receivable
  - Inventory
  - Payroll
- Productivity Tools
  - Word Processor
  - Spreadsheet
  - Data Base
  - Communications
  - Graphics
- Education
  - Tutorial
  - Skill Remediation
  - Drill and Test
  - Programmed Instruction
  - Simulations
- Entertainment
  - Shoot-em-ups
  - Strategy Games
  - Fantasy Games
  - Simulations

Utilities
    Programming Aids
    Communications
    Graphics

Looking quickly at the list above reveals that in this book we are dealing with an area of productivity tools called data bases. Before we begin construction of our data base, and subsequently use it for an amusing trivia game, we must describe and understand just what a data base is. And that's what is coming up in the next chapter.

Chapter 2

What Is a Data Base?

The term **data base** is in itself quite descriptive. A collection of information, arranged in some nonrandom and accessible order is a data base. Your telephone white pages are a data base. The phone book gives multiple iterations of the same types of information for many listees — last name, first name, address, and telephone number. *The Joy of Cooking* cookbook is a collection of recipes, each of which has ingredients, and step-by-step directions for the preparation of food. It, too, is a data base. Given just these two examples, think about what other everyday sources of information could be regarded as data bases.

To better understand what electronic (computerized) data bases consist of and do, the following analogy to a commonly used manual system of maintaining a data base may be helpful. That system is the ever present 3 x 5 filing card system. Many such index card systems are the result of our desire to collect. Once our collection reaches a significant size, we need to have control of its contents. This control can be used to trade, to sell, to insure, to value, or for any number of other activities. For whatever reason, we definitely need to control the collection's contents.

Let us suppose that we collect cassette tapes of old time radio show broadcasts, and we want to be able to share them with friends. Our collection has grown to over 600 shows, and memory alone will not suffice to summon up the exact details of each show. Our friends ask if we have any shows with Jack Benny. We know that there are two, but where? Time for an index file! Time, indeed, for a data base!

Simply writing down the information about a show, in paragraph form, for instance, quickly proves of limited use. For example:

Jack Benny show broadcast June 4, 1938. Guest stars

include Edgar Bergen and Charlie McCarthy. The show was sponsored by Lucky Strike, and is currently located in my upper-left hand desk drawer, on the Sony tape with the Red and Black label.

This card certainly has all of the information we want, but after we have finished ten cards or so, we begin to file them. How? Alphabetically, of course. But alphabetically by what criteria? For starters, let's do it by the name of the show. What is the name of the show? "T" for the, "J" for Jack, or "B" for Benny? We obviously need some standard procedures. Also, when flipping through the cards, we need to be able to find the information on the show name quickly. Let's put the show name on a separate line at the top of the card.

The next thing that we need to read on the card is the date of broadcast. Let's put it in the upper-right hand corner. And the guests . . . second line, left hand side. We are quickly designing a format for the information. Ultimately, it could end up looking like this:

Show title:	Bdcst Date:
Guests:	Sponsor:
Location:	Running time:
Additional Comments:	

At this point, let's digress for just a moment to point out another phenomenon resulting from the increasing use of microcomputers. It has been dubbed *computerphobia*, and it is often seen in the following form: the media has convinced us all that we must be "computer literate" if we are to survive economically and socially in the next decade. We also are "required," if we would be thought "good parents," to provide "computer literacy" for our children. Otherwise, perhaps we could be seen as impeding their growth and success potential as they grow up and enter a world controlled by computers. However, we sometimes feel inadequate (if not plain stupid) because we don't understand microcomputers, and if we admit it by asking questions, people will then learn the worst — we really *are* stupid.

Fortunately, this is not at all the case. Most often, it is not the *concept* which we do not understand, it is the *jargon* used to describe computers and their uses. The following exercise may help to shed light on this point: set out below are two versions

of a paragraph describing the use of our filing card system. The first uses terms with which we are all familiar and the description and concept are easily understood. The second, however, substitutes the terms which would be used to describe the same data base if it were found in a computer environment. By comparing the two, you can see that (1) you can understand concepts of microcomputer usage, and (2) you are definitely *not* stupid.

Ordinary version

Now that we have the information layout, we can begin to fill out cards for each specific show. We can then file them alphabetically by show title. After all of the cards have been filled out, we can use the newly formed card file. If, as time goes by, we change our collection, we can remove cards, change cards, or insert cards to reflect those changes. We can also decide to file them alphabetically by another bit of information on each card, such as Guest Stars, and re-sort them for location using that new category.

Computerese version

Now that we have the format, we can begin to enter data for each specific show. We can then sort the data records by show title. After all of the records have been entered, we can access the newly formed data base. If, as time goes by, we change our collection, we can delete records, modify or edit records, or add records to reflect those changes. We can also decide to sort them alphabetically by another field, such as Guest Stars, and resave all records for access using the new key field.

In general terms, then, a computerized data base does the same things as a card system. So why bother with creating and maintaining such a data base? Because a computerized system can do many other things (far more quickly and easily) that the card system can't. For example, let us suppose that we wanted to find a show that was exactly 28 minutes long. With the cards, we would have to check each card by hand, or re-sort the cards by show length, from shortest to longest or vice versa. On the computer, we can search on the part of the card (field) which has

that information until such a show is found, and then read (access) that whole record. Such sorts and searches using a variety of keystrokes are the backbone of an electronic data base.

Next, let us suppose that we also had established a dollar value for each show, and entered that onto our format in its own location (field). A sophisticated data base would also allow us to add up all of the values, thus giving us a total value for the collection at any point in time.

Finally, we shall assume that our insurance company wants limited information on each show in order to issue a policy on the collection. With the cards, we would have to hand copy or electronically copy the cards. With an electronic data base, we could design a new format for printing the data, in columns for example, and summarize all of our shows on a few pages.

In summary, an electronic data base allows for the creation of data entry and output formats, and the actual entry, storing, retrieval, editing, deletion, searching, and sorting of records.

In addition to these features, our trivia data base will be complemented by a random inquiry program. Our data will consist of questions and two word answers, and the second program will randomly select questions from data entered into the data base program itself, compare our answer, and then score us on the speed and accuracy of our replies.

Chapter 3
Where Do We Begin?

Now that we understand something of what a generalized electronic data base is and what it does, we can begin to construct our own specialized random inquiry version of a data base. The method of this book beginning with Chapter 4 will be as follows: first, define the aspect of our data base program to be dealt with; and second, to present and explain the BASIC language code which will achieve that result. As a tool for following the logical flow of both programs, we will use standard flowcharting techniques.

Such a process of program development is referred to as modular programming. Each task will have its own section of code and, when we put them all together at the end, we will have a program that meets our original design specifications.

Before we begin, however, let us review in more detail what it is that we want our program to do. First, we want to be able to enter trivia data questions and answers (in pairs). Next, we want to be able to edit (add, delete, alter) those entries until we are satisfied with the results. Finally, we want a second program to arrange for the computer to ask us those questions randomly, and give us an individual or comparative score for our efforts.

Perhaps a few comments on programming techniques and style would be helpful at this time. With regard to structure, the BASIC language can be something of a trap for the unwary. The necessity to access subroutines in other parts of our program could result in the creation of "spaghetti code" (a term meaning program code written in such a haphazard fashion that it "wanders" up and down and decreases program efficiency and legibility as it goes) if we are not careful. Top down or structured programming is much to be preferred, and it means to start at the top of the program and work out the details in a logical, sequential manner. This requires that we have a very precise idea of how the logic of the program will work before we write a

single line of program code.

The assignment of variable names within the program is also of great importance. Prepare a logical scheme for these variables by giving each variable a form which can be easily recalled. In other words, make the variable names mnemonic — a great help as we reuse the variables throughout the program.

REMark statements within the program help remind us of the function of modules, and aid other programmers who might at some later date be working with the program to understand the logic which we are using.

Finally, we cannot always assume that a user will follow our directions, and we must therefore allow for circumstances in which input might not normally be expected, or it might take an invalid form. The process of accounting for such events is called "error trapping." This process is nothing more or less than anticipating inappropriate actions by the program users, and preventing those incorrect actions from causing the program to fail, or "bomb" as we say in the trade.

We are now ready to build the skeleton of the program. Next stop, the flowchart (not Greenwich Village).

Chapter 4
Program Description

In the complete program listings which appear in Appendices A, B, C, and D, you will note that this package actually consists of four different programs. In fact, there are two different versions of two programs, all delivered on the same cassette if you bought this package as a Combo Pack. On the tape there is a tape version of the data base and the access routine, as well as a disk version* of each program which will not execute on a tape based machine, but which is meant to be loaded from tape, stored to a disk system, and then executed. For continuity and clarity we shall discuss the construct and code of the tape versions. The differences in the disk version have to do with disk input and output for the storage and retrieval of individual records created by the data base and used by the game.

One popular misconception (although *you* don't buy it for a minute!) about computers is that they can think. Computers cannot think. They can only be programmed to evaluate a certain situation according to certain guidelines, and then to perform certain calculations based on those guidelines. A specific example may be of some help here.

Let us suppose that we want to create a program which receives as input from a user his sex, height, weight, and age. Then we want to program the computer to return a message as to whether the person who input the data is underweight, at a healthy weight (for him), or overweight. Before we can begin writing BASIC code, we must define what the process will be by which the computer can output the appropriate response.

For a first pass, let us assume that we have a chart of appropriate weights for adults. We can program the computer to retain that chart, and look up (based on the user input) an appropriate

---

* Disk version program listings and variable listings for main programs and subroutines are shown in Appendices C and D.

weight based on sex, height, and age. For now, let's just assume that a program exists which will accomplish that task. We now have, in the computer's memory area, the target weight, and the actual weight of our subject. Now we come to the kind of logical branching activities which computers can be programmed for, and that lead people to believe that computers can indeed think. Actually, all that takes place are tests on the data, with selection of the correct message based on the results of those tests.

For the sake of this example, let us further assume that anyone who is plus or minus 5% of his best weight is close enough, and will receive a positive message. If his weight is less than 95% of his target weight he will get a "skinny" message, and if he is over 105% of his best weight, he will read a "fat" message.

The preceding description of program flow is wordy, awkward, and not easily understood at a glance. There must be a better way, and there is. What we have here is a sequence of tests, decisions, and actions, which can be nicely represented by a flowchart. (See flowchart on next page.)

It is now clear at a glance that we receive input from a user, compare the actual and target weights, and then choose one of three messages to print to the screen. Even in this simple example, the value of the flowcharting tool is evident. In more complex programs (such as we are describing here), flowcharting is an invaluable aid to understanding what is happening in the program. See Flowcharts 1, 2, 3, and 4 which fold out of the back of this book.

Let us now look at the heart of our data base program, using a flowchart. (Pull out Flowchart 1 for simultaneous viewing.) From the standpoint of the user, his first decision will be to load an existing data file, or create a new one.

If he chooses to load an existing file, he will be led through the process with screen messages, and if he chooses to create a new one, he will be given four more choices. These lists of choices presented on the computer screen are called menus, and a program which always has menu options available on the screen is said to be menu driven (and sometimes even "user friendly").

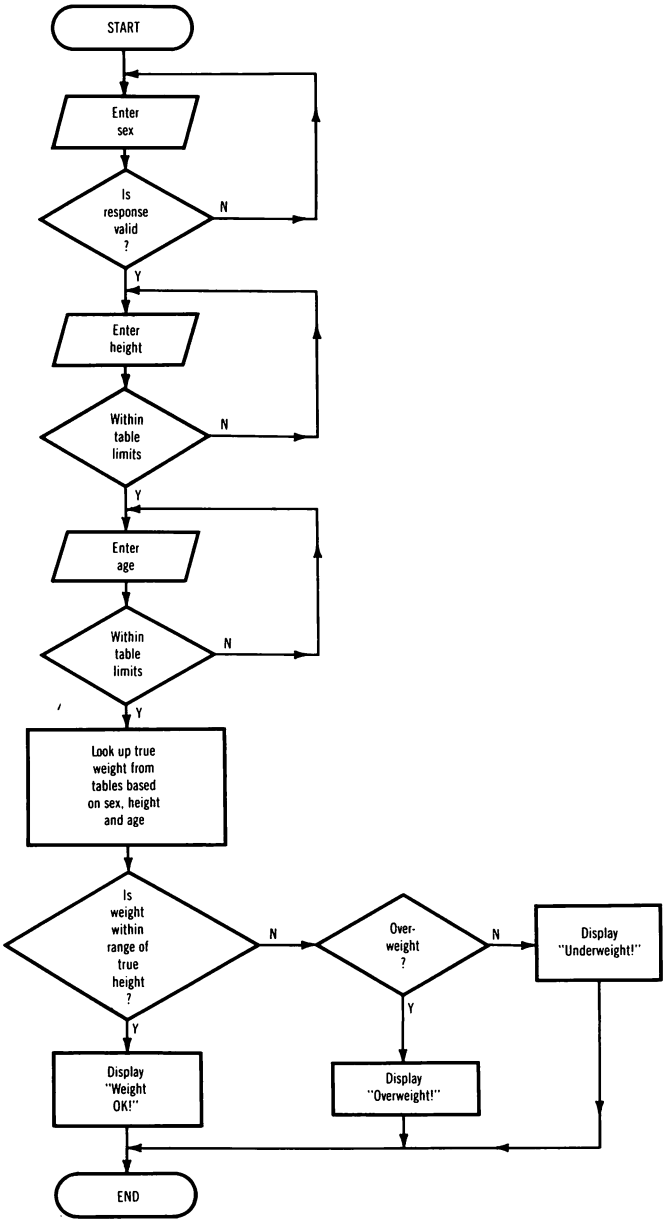


Fig. 4-1. Weight example program flowchart.

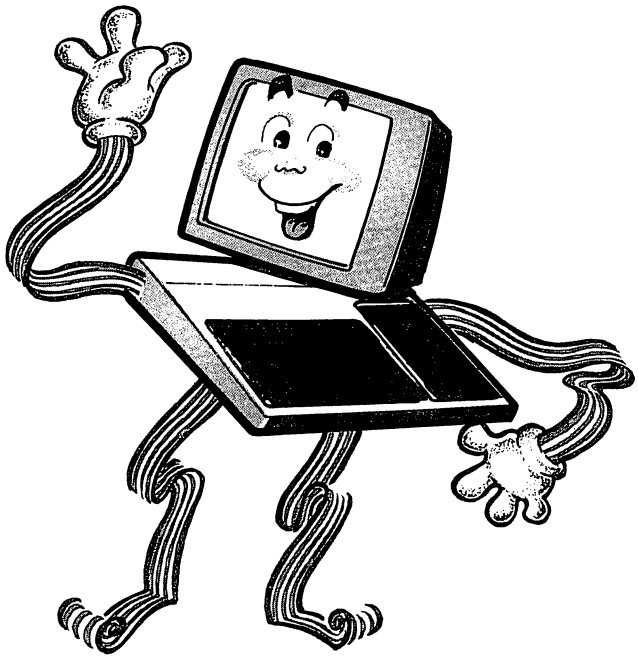


Fig. 4-2. Mr. User Friendly.

The BASIC code which will initialize our variables (they have to start somewhere) and present the first two choice menu looks like this:

```
10 REM TRIVIA DB-CASSETTE VERSION
20 OPTION BASE 1 :: DIM QA$(30)
30 CALL CHAR(128,"FF00000000000000"):: CALL  CHAR
   (129,"00007E7E7E7E0000"):: CALL CHAR
   (136,"FFFFFFFFFFFFFFFF")
40 CALL COLOR(14,13,4):: BLK$=RPT$(" ",78):: FALSE=0
   ::TRUE=NOT FALSE :: GOTO 670

670 CHNGS=FALSE :: NUMQ=0 :: CALL CLEAR :: CALL
   SCREEN(4):: DISPLAY AT(1,8)BEEP:"** TRIVIA DB **"
680 DISPLAY AT(5,2):"DO YOU WANT TO:" :: DISPLAY
   AT(8,3):"1. LOAD AN EXISTING FILE"
690 DISPLAY AT(10,3):"2. CREATE A NEW FILE" :: DISPLAY
   AT(13,3):"PLEASE SELECT (1-2)"
700 CALL KEY(0,K,S):: IF S<=0 THEN 700
710 DISPLAY AT(13,23):CHR$(K):: IF (K<49)OR(K>50)THEN
   DISPLAY AT(13,23)BEEP SIZE(1):: GOTO 700
```

The four new choices are as follows:

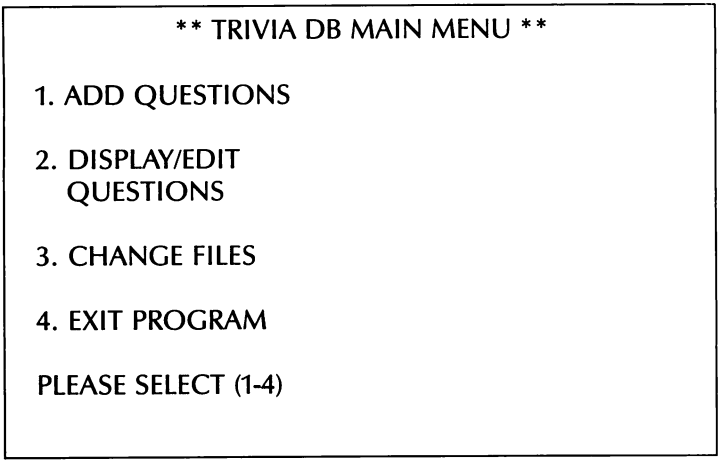


Fig. 4-3. The Trivia DB main menu.

The code which produces this main menu follows:

```
790 ARRNUM=NUMQ
800 DISPLAY AT(4,3)ERASE ALL BEEP:"** TRIVIA DB MAIN
    MENU **" :: DISPLAY AT(8,6):"1. ADD QUESTIONS"
810 DISPLAY AT(10,6):"2. DISPLAY/EDIT" :: DISPLAY
    AT(11,9):"QUESTIONS" :: DISPLAY AT(13,6):"3.
    CHANGE FILES"
820 DISPLAY AT(15,6):"4. EXIT PROGRAM" :: DISPLAY
    AT(18,6):"PLEASE SELECT (1-4)"
830 CALL KEY(0,K,S):: IF S<=0 THEN 830
840 DISPLAY AT(18,26):CHR$(K):: IF (K>48)AND(K<53)THEN
    K=K-48 :: SL=K ELSE DISPLAY AT(18,26)BEEP
    SIZE(1):: GOTO 830
850 ON K GOSUB 70,340,630,630
860 GOTO 800
```

Based upon which option is selected, there are three different subprograms which will be activated. On Flowchart 1, those choices will direct flow to B, C, or D. The following three chapters will deal with an evaluation and discussion of the flowchart for that option, and the resultant BASIC code which will achieve that flow.

Chapter 5
Adding Questions

Before reading further, pull out Flowchart 1 for simultaneous reference and see Appendix A.

Increment the counter . . . the phrase sounds like gobbledegook, but it really is quite simple. The operation ARRNUM = ARRNUM + 1 simply indicates that the next question will have a number one greater than the previous number. ARRNUM is a counter, and increment means add 1. Because we can only have up to 30 questions in one data base, we then test to see if we have exceeded that amount. If we have, then we let our user know. If not, we present a screen for input of the question and the answer which is correct.

But we are all human, and we might need to make some changes, because when we entered the question and answer, we made a mistake. We therefore offer the options of redoing the question and answer, saving it (and thus making it part of the current data file in memory only), or forgetting the whole thing and exiting the entry mode. If we choose to quit, we delete the current record, but retain the rest of the saved records that are in memory. If we save what we have done, we set the CHNGS flag (simply a marker on the status of things) to TRUE, and save the work done to memory.

Before presenting the BASIC code for this section, another thing needs to be explained. Subroutines are of great value to the programmer, because quite often a particular action is repeated throughout a program. Rather than copy the same code over and over, that action is identified as a "subroutine," and given a name. Then, when the action is required, that subroutine is invoked, or called, and many lines of code have been eliminated.

In the code for this section, please note that there are a number of subroutine calls. In most cases, the name of the routine, and the attendant code, are straightforward and sim-

ple. A complete listing of the subroutines in this program, by name, is contained in Appendix A, and should be referred to as each is invoked.

The code for adding questions follows:

```
70 CALL CLEAR :: DE=FALSE
80 ARRNUM=ARRNUM+1 :: IF ARRNUM>30 THEN ARRNUM=30 ::
   CALL FULL :: RETURN
90 TT$="**  ADDING QUESTIONS  **" :: CALL  SCRFORM
   (ARRNUM,TT$):: CALL ADDCMDS
100 Q$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE :: CALL
   GETCHAR(7,2,3,Q$,FLAG,SV,EX)::IF FLAG=1 THEN CALL
   ERASEQUEST :: GOTO 100
110 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE IF
   FLAG=2 AND Q$="" THEN 100
120 Q$=SEG$(Q$&BLK$,1,78):: CALL WRAP(Q$,BLK$)
130 A$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE :: CALL
   GETCHAR(17,2,1,A$,FLAG,SV,EX)
140 IF FLAG=1 THEN CALL ERASEQUEST :: CALL ERASEANS ::
   GOTO 100
150 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE
   GOSUB 60 :: IF ERR THEN CALL ERRMSG :: GOTO 130
160 IF FLAG=2 THEN GOSUB 50 :: CHNGS=TRUE :: GOTO 80
170 CALL HCHAR(23,31,129):: CALL BEEP
180 CALL KEY(0,K,S):: IF S<=0 THEN 180
190 IF K=146 THEN GOSUB 230 :: CALL ADDCMDS :: GOTO
   170
200 IF K=147 THEN FLAG=2 :: GOTO 160
210 IF K=152 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE CALL
   BEEP :: GOTO 180
```

There are two additional sections of code that are invoked within this portion of the program. First, it is desirable to eliminate the articles "a," "an," and "the" from answers, since we are limited to two word answers, and someone might well delete them, but get the rest of the answer correct. We therefore have a routine called Parseans which deletes them. In line 60, when PARSEANS is called, you will note that there are two variables in parentheses following the call. These are called arguments and are important in the calling process. Argument is, again, a confusing term in this context. Basically, what it means is that the subroutine called PARSEANS requires two pieces of information before it can perform its function, and that it requires them in a particular order. In this case it needs to have a variable called ANS$ followed by another numerical variable called ERR. For lists of variable names for the programs in this book, see, as appropriate, Appendix A, B, C, or D.

Secondly, there is a portion of code which handles the storing of questions and answers:

The call for the parsing routine looks like this:

```
60 ANS$=A$ :: ERR=FALSE :: CALL PARSEANS(ANS$,ERR)::
    RETURN
```

That, then, comprises the code for the ADD RECORDS section of the program. But, as noted earlier, nobody is perfect, and even after we have entered what we think is correct information, we sometimes will have to change it. Time to look to the editing process, and that is discussed in the next chapter.

Chapter 6
Editing Questions

This module is undoubtedly the most complex one in the data base program. Pull out Flowchart 1 for simultaneous reference, refer to Appendix A, and let's look at the logic.

First we test to make sure that our location in the list of questions is not before record number 1. If we are, we advise the user, and return to the main menu. If not, we assign our current position within the file to the variable ARRPOS. We then display the current record question and answer, and offer it up for modification. We have several options:

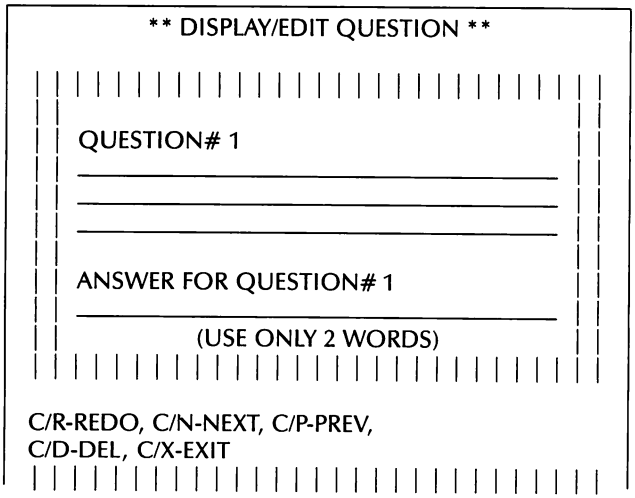


Fig. 6-1. Display/edit screen.

The first option is to redo the question and/or the answer. Look at the rest of the flowchart, and you can see that we have even more options. At this point we can change the question, the answer, both, or neither. If we wish to change the question and/or the answer, we first erase the old entry, and then replace

it with the new information.

Notice that, as before, there is a flow option for "invalid responses." These are our error trapping routines. Now let us return to option C on Flowchart 1. The second option is to look at the next record. ARRPOS is incremented by 1, and then we test to make sure that we have not passed all of the existing records in the file. If we pass both tests, it's back to display and editing. If not, we present an appropriate message, and return to the previous menu.

Likewise, we can also look at the previous record. We decrement the record position, check for the beginning of the file, and then edit that record. We may simply opt to delete a record altogether. We check to make sure that is the requested action (once it's gone, it's gone), and then perform the action. Finally, we can choose to exit this module and return to the main menu.

The code for this section follows. Remember, subroutines can be found in Appendix A.

```
340 IF ARRNUM>0 THEN 370
350 CALL CLEAR :: DISPLAY AT(6,2)BEEP:"THERE ARE NO
    QUESTIONS"
360 DISPLAY AT(8,2):"AND ANSWERS IN MEMORY TO" ::
    DISPLAY AT(10,2):"DISPLAY/EDIT." :: CALL ENTER ::
    RETURN
370 ARRPOS=1 :: DE=TRUE
380 TT$="* DISPLAY/EDIT QUESTION *" :: CALL
    SCRFORM(ARRPOS,TT$)
390 DISPLAY AT(7,2):SEG$(QA$(ARRPOS),1,26):: DISPLAY
    AT(9,2):SEG$(QA$(ARRPOS),27,26)
400 DISPLAY AT(11,2):SEG$(QA$(ARRPOS),53,26):: DISPLAY
    AT(17,2):SEG$(QA$(ARRPOS),79,26)
410 CALL EDITCMDS
420 CALL KEY(5,K,S):: IF S<=0 THEN 420
430 IF K=132 THEN 530 ELSE IF K=142 THEN
    ARRPOS=ARRPOS+1 :: GOTO 470
440 IF K=144 THEN ARRPOS=ARRPOS-1 :: GOTO 500
450 IF K=146 THEN TOT=ARRNUM :: ARRNUM=ARRPOS :: GOSUB
    220 :: GOSUB 50 :: ARRNUM=TOT :: GOTO 380
460 IF K=152 THEN RETURN ELSE CALL BEEP :: GOTO 420
470 IF ARRPOS<=ARRNUM THEN CALL NUMDISP(ARRPOS):: GOTO
    390
480 DISPLAY AT(23,1)BEEP:"THERE ARE NO MORE
    QUESTIONS!":: DISPLAY AT(24,1)SIZE(28)
490 CALL DELAY(600):: CALL EDITCMDS :: CALL BEEP ::
    ARRPOS=ARRPOS-1 :: GOTO 420
500 IF ARRPOS>=1 THEN CALL NUMDISP(ARRPOS):: GOTO 390
510 DISPLAY AT(23,1)BEEP:"CAN'T GO BELOW RECORD #1!"
    ::DISPLAY AT(24,1)SIZE(28)
```

```
520 CALL DELAY(600):: CALL EDITCMDS :: CALL BEEP ::
    ARRPOS=ARRPOS+1 :: GOTO 420
530 DISPLAY AT(23,1)BEEP:"DELETE THIS RECORD? (Y/N)"
    ::DISPLAY AT(24,1)SIZE(28)
540 CALL KEY(0,K,S):: IF S<=0 THEN 540
550 CALL HCHAR(23,30,K):: IF K<>78 AND K<>89 AND
    K<>110 AND K<>121 THEN CALL BEEP :: GOTO 540
560 IF K=78 OR K=110 THEN 410 ELSE IF ARRNUM-1=0 THEN
    QA$(ARRPOS)="" :: ARRNUM=0:: GOTO 350
570 IF ARRPOS=ARRNUM THEN QA$(ARRNUM)="" ::
    ARRPOS=ARRPOS-1 ELSE QA$(ARRPOS)=QA$(ARRNUM)
580 ARRNUM=ARRNUM-1 :: CHNGS=TRUE :: IF ARRNUM<1 THEN
    350 ELSE CALL NUMDISP(ARRPOS):: GOTO 390
```

Chapter 7
Saving and Exiting

Now that we have finished with the data base, we need to leave the program cleanly, and allow for file updates as needed. Look at Flowchart 1, in the pullout section, as well as Appendix A while we evaluate the final portions of the data base code.

First we check to make sure that some changes have been made to the active data base. If so, then you have a chance to save the file from memory to tape. Otherwise, we clear the screen to end the program.

If changes have been made, then we ask if we want to save the changes. If so, we do; if not, we return to the main menu. All of these checks are designed to allow a user to "back out of" an erroneously chosen option, without damaging the integrity of the data.

The code for these two sections follows:

```
590 CALL SAVEMSG :: CALL CLEAR
600 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 128
610 PRINT #1:NUMQ :: FOR I=1 TO NUMQ :: PRINT
    #1:QA$(I):: NEXT I
620 CLOSE #1 :: RETURN
630 IF NOT CHNGS THEN IF SL=4 THEN CALL CLEAR :: END
    ELSE 670
640 NUMQ=ARRNUM :: YFLG=FALSE :: CALL SAVECHNGS(YFLG)
650 IF NOT YFLG THEN IF SL=4 THEN CALL CLEAR :: END
    ELSE 670
660 GOSUB 590 :: GOTO 670
```

This completes the code explanations for the data base itself. Next we shall look at the random inquiry and scorekeeping program (the game), using the same techniques employed in the preceding analysis.

Chapter 8
Beginning the Trivia Game

We can now begin an analysis of the second program in this package, the *Trivia Game,* which will utilize the data base created by the first program. Pull out Flowchart 2 for simultaneous viewing, see Appendix B, and we shall begin an evaluation of the flowchart and attendant code.

As was the case with the data base program, we must first initialize the variables. We then display the introduction screen. This code appears as follows:

```
10 REM TRIVIA GAME - CASSETTE VERSION
20 CALL CLEAR :: CALL SCREEN(4):: OPTION BASE 1 :: DIM
    QA$(30),RQN(30),N(30),ND(8),NT(4)
30 BLK$=RPT$(" ",26):: FALSE=0 :: TRUE=NOT FALSE ::
    CALL CHAR(128,"FF0000000000000000"):: CALL
    CHAR(129,"00007E7E7E7E0000")
40 CALL CHAR(136,"FFFFFFFFFFFFFFFF"):: CALL
    COLOR(14,13,4):: GOTO 200

200 DISPLAY AT(6,4)BEEP:"WELCOME TO THE TI-99/4A" ::
    DISPLAY AT(10,6):"**  TRIVIA GAME  **" :: CALL
    ENTER
```

Next, we must load the data files from tape (or disk) prior to starting the game. We first ask for the number of cassette files. If there is only one file, we will use all of the questions. If there is more than one file, then we must choose only selected questions from each tape. The code for this activity follows (remember, subroutines are called by name, and can be found in Appendix B):

```
210 DISPLAY AT(8,2)ERASE ALL BEEP:"HOW MANY CASSETTE
    FILES" :: DISPLAY AT(10,2):"DO YOU HAVE? (1-10)"
220 ACCEPT  AT(10,21)VALIDATE(NUMERIC) SIZE(3)
    BEEP:NUMFILES :: IF NUMFILES<1 OR NUMFILES>10 THEN
    220
230 IF NUMFILES=1 THEN 380
240 DISPLAY AT(8,2)ERASE ALL BEEP:"PLEASE WAIT...I'M
    WORKING" :: RANDOMIZE :: FOR I=1 TO 30
```

```
250 WN=INT(RND*NUMFILES*30)+1 :: ERR=FALSE :: CALL
    CKNUM(WN,RQN(),ERR,30):: IF ERR THEN 250 ELSE
    RQN(I)=WN
260 NEXT I :: P=1 :: Q=30 :: T0=0
270 IF P>=Q THEN 360
280 V=RQN(P):: I=P :: J=Q+1
290 J=J-1 :: IF RQN(J)>V THEN 290
300 I=I+1 :: IF RQN(I)<V AND I<75 THEN 300
310 IF J>I THEN T=RQN(I):: RQN(I)=RQN(J):: RQN(J)=T ::
    GOTO 290
320 RQN(P)=RQN(J):: RQN(J)=V
330 IF (J-P)<(Q-J)THEN N(T0+1)=J+1 :: N(T0+2)=Q :: Q=J-
    1 :: GOTO 350
340 N(T0+1)=P :: N(T0+2)=J-1 :: P=J+1
350 T0=T0+2 :: GOTO 270
360 IF T0<>0 THEN Q=N(T0):: P=N(T0-1):: T0=T0-2 ::
    GOTO 270
370 FOR I=1 TO 30 :: N(I)=0 :: NEXT I
380 DISPLAY AT(2,2)ERASE ALL BEEP:" THIS IS GOING TO
    TAKE A" :: DISPLAY AT(4,2):"WHILE, FOR ME TO LOAD
    THE"
390 DISPLAY AT(6,2):"QUESTIONS AND ANSWERS FROM" ::
    DISPLAY AT(8,2):"YOUR CASSETTE."
400 DISPLAY AT(10,2):"  PLEASE BE PATIENT AS I" ::
    DISPLAY AT(12,2):"LOAD THE RANDOM QUESTIONS"
410 DISPLAY AT(14,2):"FROM YOUR CASSETTE FILES." ::
    CALL ENTER :: CN=1 :: CTR=1:: RECCTR=0
420 DISPLAY AT(20,2)ERASE ALL BEEP:"INSERT CASSETTE
    #";CN
430 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
440 INPUT #1:NUMQ :: FOR I=1 TO NUMQ :: INPUT
    #1:QA$(CTR)
450 IF NUMFILES=1 THEN RQN(CTR)=I :: CTR=CTR+1 :: GOTO
    470
460 IF I+RECCTR=N(CTR)THEN CTR=CTR+1
470 NEXT I :: RECCTR=RECCTR+NUMQ :: CN=CN+1 :: CLOSE
    #1:: IF CN<=NUMFILES THEN 420
```

Notice that, since this process is often time consuming, we display a "pacifier message" during the process to assure users that things are proceeding properly. Finally, we must "shuffle" the questions, so that the order is not predictable.

After we have shuffled the questions (and their corresponding answers), we will pick double and triple point questions. Then we will find out who is playing, by asking for the number of players and their names. We are now ready to begin the game, so we display the screen form for the game, and move onto the next chapter, where we will look at the interrogation, evaluation, and scorekeeping functions. The shuffling and selecting of double and triple point questions is coded as follows:

```
480 DISPLAY AT(6,2)ERASE ALL BEEP:"PLEASE WAIT..." ::
    DISPLAY AT(10,2):"SCRAMBLING QUESTIONS"
490 RANDOMIZE :: NUMQ=RECCTR :: IF NUMQ>30 THEN
    NUMQ=30
500 NUMDBL=INT(NUMQ*.25)+1 :: NUMTPL=INT(NUMQ*.10)+1
510 FOR I=1 TO NUMQ
520 WN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(WN,N(),ERR,NUMQ)
530 IF ERR THEN 520 ELSE N(I)=WN
540 NEXT I :: FOR I=1 TO NUMDBL
550 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(QN,ND(),ERR,8)
560 IF ERR THEN 550 ELSE ND(I)=QN
570 NEXT I :: FOR I=1 TO NUMTPL
580 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(QN,NT(),ERR,4):: IF ERR THEN 580
590 CALL CKNUM(QN,ND(),ERR,8):: IF ERR THEN 580
600 NT(I)=QN :: NEXT I
610 DISPLAY AT(10,2)ERASE ALL BEEP:"NUMBER OF PLAYERS
    (1-4)"
620 CALL KEY(0,K,S):: IF S<=0 THEN 620
630 CALL HCHAR(10,28,K):: IF K<49 OR K>52 THEN CALL
    BEEP :: CALL HCHAR(10,28,32):: GOTO 620
640 NP=K-48 :: FOR I=1 TO NP :: DISPLAY AT(10,2)ERASE
    ALL:"PLAYER#";I;"NAME:"
650 ACCEPT AT(10,18)VALIDATE(UALPHA)BEEP
    SIZE(11):PN$(I):: IF PN$(I)="" THEN 650
660 NEXT I :: DISPLAY AT(1,5)ERASE ALL BEEP:"**
    TRIVIA GAME  **" :: CALL HCHAR(2,3,136,29)
```

Now, on to the actual play of the game.

Chapter 9
The Game Itself

For this section of the program, pull out Flowchart 2 and see Appendix B for reference. We have displayed the format, and now we must get one of the random questions previously loaded into memory. We check to find out whether the question has been randomly designated as a double or triple value question.

Now that we know what the question is worth in point value, we start the timer, and wait for the user to enter his answer. As time passes, the value of the question decreases. Once the **ENTER** key is pressed, we have to check for the correctness of the answer. If the answer is correct, we go to the next player, and display his current score, if there is another player remaining. We then pick the next question, and display it.

Once the questions have been used up, we must display the final score. This done, we ask for a replay, and either start over or end the game with a pleasant message. These code sections follow:

```
720 PLYR=1 :: ARRLOC=1 :: FOR I=1 TO NP :: NQ(I)=0 ::
    NC(I)=0 :: SC(I)=0 :: NEXT I
730 TIM=0 :: CALL TIMER(TIM):: GOSUB 50 :: DISPLAY
    AT(5,13)SIZE(2):ARRLOC
740 DISPLAY AT(7,2):SEG$(QA$(N(ARRLOC)),1,26)::
    DISPLAY AT(9,2):SEG$(QA$(N(ARRLOC)),27,26)
750 DISPLAY AT(11,2):SEG$(QA$(N(ARRLOC)),53,26)::
    STPLYR=PLYR
760 FOR I=1 TO NUMDBL :: IF ND(I)=N(ARRLOC)THEN
    DBL=TRUE :: GOTO 790
770 NEXT I :: DBL=FALSE :: FOR I=1 TO NUMTPL :: IF
    NT(I)=N(ARRLOC)THEN TPL=TRUE :: GOTO 790
780 NEXT I :: TPL=FALSE
790 DISPLAY AT(15,2)SIZE(26):: DISPLAY
    AT(12,9)SIZE(15)
800 IF DBL THEN CALL BEEP :: DISPLAY AT(12,10):"DOUBLE
    VALUE" :: CALL BEEP
810 IF TPL THEN CALL BEEP :: DISPLAY AT(12,10):"TRIPLE
    VALUE" :: CALL BEEP
```

```
820 TU=FALSE :: ANS$="" :: QUIT=FALSE :: CALL
    GETANS(ANS$,15,2,26,QUIT,TIM,TU)::POINTS=25-TIM ::
    IF QUIT THEN 940
830 IF TU THEN 860 ELSE IF DBL THEN POINTS=POINTS*2
840 IF TPL THEN POINTS=POINTS*3
850 CALL PARSEANS(ANS$):: ANS$=SEG$(ANS$&BLK$,1,26)::
    CA$=SEG$(QA$(N(ARRLOC)),79,26):: CALL
    CONVERTUPPER(CA$):: IF CA$=ANS$ THEN 910
860 CALL SOUND(700,-3,0):: CALL DELAY(200)::
    NQ(PLYR)=NQ(PLYR)+1
870 PLYR=PLYR+1 :: IF PLYR>NP THEN PLYR=1
880 IF PLYR<>STPLYR THEN TIM=13 :: CALL TIMER(TIM)::
    GOSUB 50 :: GOTO 790
890 DISPLAY AT(14,2)BEEP:"THE CORRECT ANSWER IS:" ::
    DISPLAY AT(15,2):SEG$(QA$(N(ARRLOC)),79,26)
900 CALL DELAY(500):: DISPLAY AT  (14,2)SIZE(26):
    "ANSWER:" :: DISPLAY AT(15,2)SIZE(26):: GOTO 920
910 CALL SOUND(700,-1,0):: SC(PLYR)=SC(PLYR)+POINTS ::
    NC(PLYR)=NC(PLYR)+1 :: NQ(PLYR)=NQ(PLYR)+1 ::
    GOSUB 50  :: CALL DELAY(300)
920 ARRLOC=ARRLOC+1 :: IF ARRLOC>NUMQ THEN CALL
    DELAY(200):: GOTO 70 ELSE PLYR=STPLYR+1 :: IF
    PLYR>NP THEN PLYR=1
930 DBL=FALSE :: TPL=FALSE :: GOTO 730


70 CALL SOUND(300,-1,0):: CALL SOUND(300,-1,0)::
    DISPLAY AT(1,10)ERASE ALL:"FINAL STATS"
80 DISPLAY AT(5,2):"NAME         SCORE   NC/NQ" :: CALL
    HCHAR(6,4,128,29)
90 IF NP=1 THEN DISPLAY AT(8,2):PN$(1):: DISPLAY  AT
    (8,14):SC(1):: DISPLAY AT(8,22):NC(1);"/";NQ(1)::
    GOTO 160
100 FOR I=1 TO NP :: PN(I)=I :: NEXT I :: FOR J=1 TO
    NP-1 :: J1=J
110 FOR K=J+1 TO NP :: IF SC(J1)<SC(K)THEN J1=K
120 NEXT K :: IF J1<>J THEN T=SC(J):: SC(J)=SC(J1)::
    SC(J1)=T :: T=PN(J):: PN(J)=PN(J1):: PN(J1)=T
130 NEXT J
140 FOR I=1 TO NP :: DISPLAY AT(6+I,2):PN$(PN(I))::
    DISPLAY AT(6+I,14):SC(I)
150 DISPLAY AT(6+I,22):NC(PN(I));"/";NQ(PN(I)):: NEXT
    I
160 IF QUIT THEN CALL DELAY(800):: GOTO 190
170 DISPLAY AT(22,2)BEEP:"ANOTHER GAME? (Y/N)" ::
    YFLG=FALSE :: CALL GETYN(YFLG)
180 IF YFLG THEN DISPLAY AT(10,2)ERASE ALL:"ONE MOMENT
    PLEASE..." :: RUN 10
190 DISPLAY AT(10,2)ERASE ALL:"HAVE A NICE DAY!" ::
    END
```

Once the game has begun, we may wish to quit (when the score is terribly one-sided, for example). In order to do this, we may enter **Ctrl-Q** to call the following subroutine:

```
940 DISPLAY AT(23,6)SIZE(20)BEEP:"ARE YOU SURE? (Y/N)"
    :: YFLG=FALSE :: CALL GETYN(YFLG)
950 IF YFLG THEN 70 ELSE DISPLAY
    AT(23,6)SIZE(20):"CTRL-Q TO QUIT" :: GOTO 820
```

Notice that, as always with an ending decision, we check for a confirmation.

This completes the code for the actual game. When we put it all together, it should (and does) work. In the next two chapters, we shall document the actual use of these two programs.

Chapter 10
Using the Data Base

As noted earlier, the *Trivia Data Base* is the program that is used to store your questions and answers. This chapter demonstrates how easily the data base program can be to use. (Just a note before we begin. You will need to have blank cassettes handy at all times and they should be labelled in some organized way. For example, Trivia Files #1, Trivia Files #2, etc., or even Cassette #1, Cassette #2, etc., will work. The reason for the blank cassettes is that the program (for memory convenience) stores 30 questions and answers on each cassette and if you have more than 30 questions you'll need to store the rest on a different cassette. Another little note for you, DB stands for data base.)

The data base is totally menu driven. This means that you are able to select the process that you want to perform on the data base. Once the program is run, you are presented with an initial menu.

```
** TRIVIA DB **

DO YOU WANT TO:

1. LOAD AN EXISTING FILE

2. CREATE A NEW FILE


PLEASE SELECT (1-2)
```

Fig. 10-1. The Trivia DB initial menu.

From this menu you select whether you want to load and work on a Trivia File that has already been created with this program, or create a new Trivia File. Because this is your first time using the data base, you'll want to select option #2, CREATE A NEW FILE, but before doing that let me explain each option.

LOAD AN EXISTING FILE

This option allows you to load a Trivia File into memory so you are able to add new questions to the file or change questions and answers. Once this option is chosen, a message is displayed saying to insert your cassette file into your tape player. Then follow the directions that are given by the TI-99/4A computer. If for any reason or by chance an error occurs while loading in your file, the program takes you to the TRIVIA DB MAIN MENU (Fig. 10-2) with no questions in memory at the time.

CREATE A NEW FILE

This option allows you to create a new file. Any file can contain at most 30 questions and answers. It is because of memory and speed consideration that we arrived at the limit of 30. Go ahead and select this option now, and then you are presented with the TRIVIA DATA BASE MAIN MENU.

```
** TRIVIA DB MAIN MENU **

1. ADD QUESTIONS

2. DISPLAY/EDIT
   QUESTIONS

3. CHANGE FILES

4. EXIT PROGRAM

PLEASE SELECT (1-4)
```

Fig. 10-2. The Trivia DB main menu.

Again, let's look at each of the four selections given in Fig. 10-2.

1. ADD QUESTIONS

This is the first thing that you need to do when creating a new file. This selection allows the user to enter new questions and answers to the trivia file. The current question number is always displayed while you are adding new questions to let you know how many questions are in the file. The screen for adding the questions and answers is shown in Fig. 10-3.

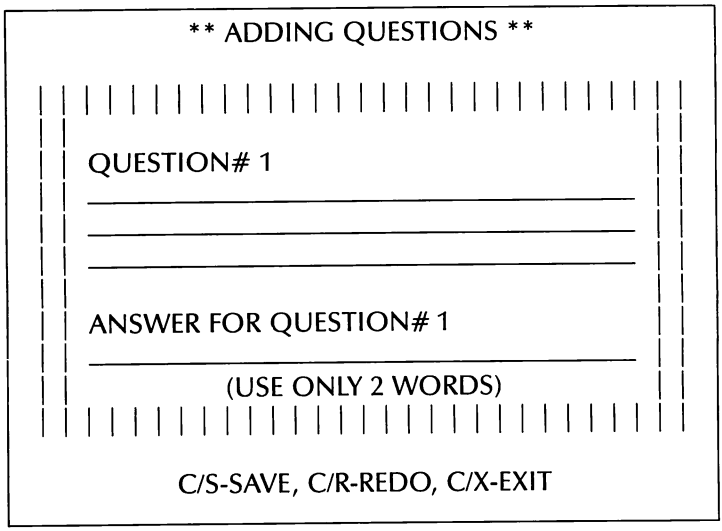


Fig. 10-3. Adding questions screen.

First take a look at the bottom of the screen in Fig. 10-3. The bottom of every screen contains valuable information. This screen contains only three major keys. They are C/S, C/R, C/X. The C stands for the **CTRL** key. So by pressing the **CTRL** key and the **S** simultaneously, the current question and answer is saved into memory. C/R allows you to redo or change the questions and answer. If the cursor (the little black rectangle that appears on the question field) is on the last line of the screen, by the three major keys, then C/R generates another display asking what you want to change. You can then choose to edit the question, answer, or both. If you press C/R while the cursor is on

either the question field or the answer field, then you must reenter both the question and the answer. The data base doesn't allow you to enter a blank question or a blank answer. Also, you may press any one of the keys displayed at the bottom of the screen at any time. By pressing the C/X (**CTRL** key plus the **X** key) you exit the ADD QUESTION routine and it also erases anything that is on the screen at the time of exiting. You are then returned to the TRIVIA DB MAIN MENU (Fig. 10-2).

Now you can enter your first question. When you are finished with your question press the **ENTER** key to move the cursor down to the answer field. Don't worry about how your question looks while you are typing, because as soon as you are finished and you press the **ENTER** key, the question gets reformatted (words are wrapped around to the next line) to look better. The cursor then moves down to the ANSWER field and waits for the answer. The answer cannot be more than two words. If it is, the computer beeps and tells you to reenter the answer. Once the user has typed in an answer then you can use C/S to save it or press **ENTER** to move the cursor down to the command line at the bottom of the screen. Again you may press any one of the three command keys at any time. Once the question and answer have been saved you are then given another blank screen to add more questions and answers.

When the cursor is on the same line as the command keys, you then have to press one of the three keys displayed. If the user presses C/S the record is saved to memory and the array counter is incremented and another blank record is displayed. If the user presses C/X, the current question and answer that is on the screen is erased and the user is returned to the TRIVIA DB MAIN MENU (Fig. 10-2). If you press C/R, you are then given the REDO command line presented at the bottom of the screen (Fig. 10-4).

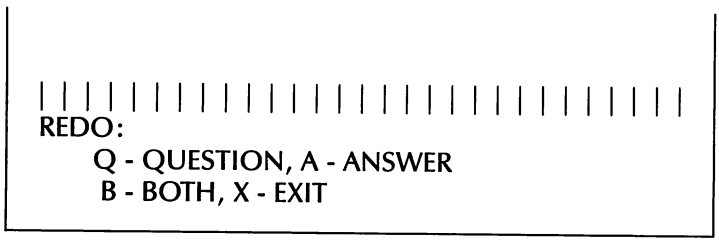


Fig. 10-4. REDO command line.

Once at this point, you can then select which part you want to redo. Either the question, the answer or both at once. When you are finished editing the record, press X to exit and return you to the C/S, C/R, etc. commands. You will still need to save the record when you are through editing it.

2. DISPLAY/EDIT QUESTIONS

This routine allows you to display as well as edit (make changes) to any question or answer in the file. When you press 2 from the Main Menu, you are presented with the DISPLAY/EDIT SCREEN (Fig. 10-5).

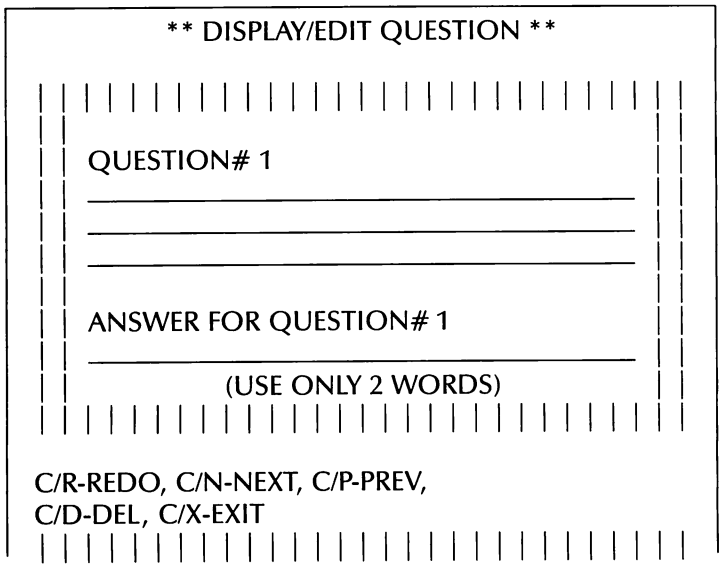


Fig. 10-5. Display/edit screen.

Again, as you notice in Fig. 10-5, at the bottom of the screen there is another command line. Here there are five command keys to use. C/R (remember that the C stands for the **CTRL** key and the other letter stands for itself) displays two rows of commands that allow the user to change the question, the answer, or both as in Fig. 10-4. You can then choose what you want to do. C/N displays the next question and answer. C/P displays the previous question and answer. C/D asks you if you are sure that this

is the question and answer that you want to delete. If you press **Y** to the prompt, the question and answer is deleted from your file. C/X exits this routine and takes you to the TRIVIA DB MAIN MENU.

3. CHANGE FILES

This selection allows you to quit working on the file currently in memory and load in a different file. If there were any changes made to the file in memory, then you are asked if you wish to save the changes made to the file. You are prompted with this question: SAVE CHANGES? (Y/N). If you press **N** everything (including the changes) will be thrown away. If you press **Y** then you are prompted to insert a cassette in the tape recorder and then follow the instructions as they are presented on the screen. Once the file in memory has been saved or erased, you are then returned to the very first screen (Fig 10-1) and then you choose from that screen what you want to do.

4. EXIT PROGRAM

This selection allows you to exit the *Data Base* program. Again, if there is any file in memory and changes were made (addition or editing) to the file, then you are asked if you wish to save the file that is currently in memory. If you elect not to save the file in memory, then everything in memory is thrown away. *Always exit the program through this selection. If you don't then none of the changes or new questions can be saved.*

Now that you have an understanding of how to enter and edit questions try putting some questions and answers in a file so that you can have some questions and answers with which to play the *Trivia Game*. If you really can't wait to play the game, then go ahead and load the sample questions.

Before we exit this chapter, let me inform you (those of you who have a disk drive) that there is a disk version of the *Data Base* and a disk version of the *Trivia Game*. This next paragraph is only for those people who have a disk drive with their TI-99/4A. I want to explain the differences between the two versions.

The major difference between the two versions (the cassette and the disk) is the I/O routines. I/O stands for input and output. This is just the way the program stores the questions and

answers and the way they are loaded into memory. The disk version keeps only one question and answer in memory at a time. The cassette version holds 30 at once. As with the cassette version, you should keep one file per disk, where a file can contain up to 710 questions and answers. Don't place a trivia file on a disk that already has other files. You might lose some questions and answers later on when the disk starts to get full. The rest of the disk version is the same as the cassette version. You are presented with the same menus, same screens, etc., as the cassette version. Again remember to exit the disk version through #4. If you don't you stand a good chance of losing some information.

NOTE ON USING THE SAMPLE FILE

There are a few mistakes in the sample file. They have been made on purpose to provide you with a file on which to practice using the data base. Remember to save the new file when you are finished.

Chapter 11
Playing the Trivia Game

In the last chapter you learned how to enter questions and answers into your Trivia Data Base File. This chapter uses those questions that you entered into the file and allows you to play a trivia game. For those of you who did not store any questions and answers using the data base, you may use the sample file that is located after the disk version of the Trivia Game on Side 1 of the tape.

INSTRUCTIONS

The game itself is much like any other trivia game. You are asked random questions and you have to answer them. The object of the game is to gain as many points as possible by answering random questions. The faster you are able to answer a question, the more points you will receive. The point value for each question starts at 25 points, but as time goes by the point value decreases. So, you see, it is to your benefit to answer the questions as fast as possible. There will be times when a question will be worth DOUBLE points and even TRIPLE. One major note: *the answer you type in must match the correct answer exactly;* otherwise, it will not count as the correct answer. The game can be played by one to four players. If a player misses the question, then the next player in line gets a chance to answer the question, although the point value is reset to 12 points instead of the original value of 25. Once everyone has a chance to answer the question and no one answers it correctly, the correct answer is displayed. Every time someone gets the correct answer a high beeping sound is made. When someone misses, they hear a lower beeping sound. The game keeps track of each player's number, his/her name, total points scored, the number of questions each player answers correctly, and the number of questions each player had a chance to answer. At the end of the

game, the players' scores are shown in descending (from highest to lowest) order.

RUNNING THE GAME

The first screen you see is the introduction screen (Fig. 11-1). It welcomes you to the *Trivia Game*. Next, you are asked how many cassette files you have. The game allows you to use up to ten different cassette files from which to have your questions selected. Although, you are only allowed a total 30 questions in memory due to memory limitations on the TI-99/4A, the process of loading in random questions takes time. The program (if more than one file is used) will generate enough random numbers to fill memory and then go through all your files and load in selected questions. Once the questions and answers have been loaded into memory, then they are scrambled again. A percentage of the questions is selected to be worth DOUBLE the point value and even a few of the questions are selected for TRIPLE value. Once all the questions have been scrambled, you are asked how many people plan to play and then their names are entered. Now, the game is ready to begin.

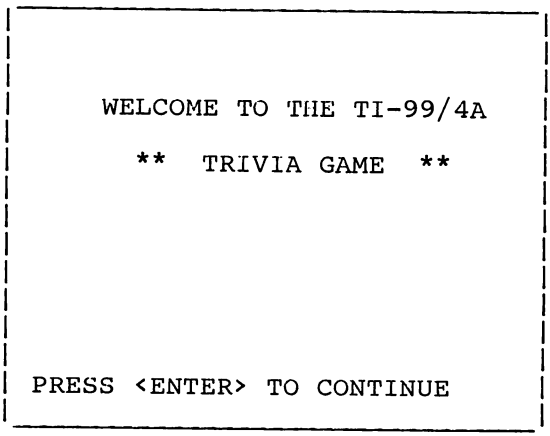


Fig. 11-1. The introduction screen.

THE START

After everyone has entered his/her name, then the game

screen is displayed (Fig. 11-2). Notice in the upper-right hand corner of the boxed-in screen, there is a timer. This timer also shows how much the question is worth. As time goes by, this timer decreases. When you finally press **ENTER** after entering your answer, the timer freezes at the point value shown. This is the amount of points that you will receive if your answer is correct. At the bottom of the screen is the status of the current player. It shows the player number, name, score, the number of correct answers (NC) and the number of questions (NQ) that that player had a chance to answer. Once you press **ENTER**, the program then checks your answer against the correct answer. If you are right, then you receive the points that are shown in the upper-right hand corner. Otherwise, the next player gets a chance to answer it. The correct answer is displayed after everyone has a chance to answer the question, and they all miss.

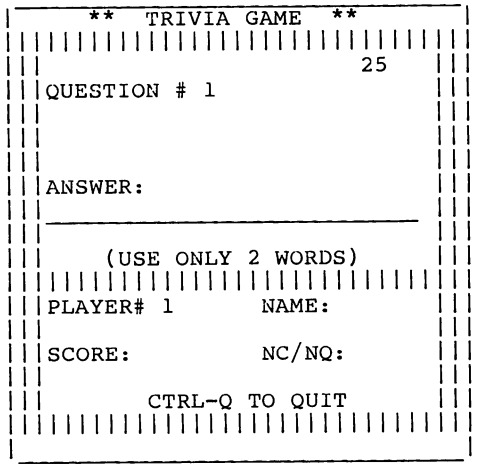


Fig. 11-2. The game screen.

EXITING THE GAME

The game normally ends when all the questions have been asked. Then there is a status screen displayed that shows all the players' scores in descending (highest to lowest score) order. You are then asked if you wish to play another game. If you don't, the program exits. Otherwise, you start all over again with loading in another file, etc.

If you get tired before all the questions have been asked, just

press **CTRL-Q** (hold down the **CTRL** key while simultaneously pressing the **Q** key) and the program will ask if you are sure you want to exit. If so, then all the players' scores are shown with the highest score shown first. If you do not wish to exit, then the game continues with the same question and the same point value that the question was at before you pressed **CTRL-Q.**

Disk Version (for those who have a disk drive attached to the TI-99/4A.). The disk version is the same as the cassette version. The only exception is that you can have more questions in memory at once, especially if you have the extended memory cartridge for your expansion box. In the disk version, you can have 30 questions in memory at once. The disk version loads your random questions all from one file (disk), which can hold up to 710 questions. The game and screen designs, as well as how the points are kept, etc., are exactly the same as the cassette version.

The main point behind the *Trivia Game* is for you to enjoy the game, while at the same time, you can be learning new and different trivia questions. Have fun!

Chapter 12
In Conclusion

Well, that's it. We have provided sample questions after the disk version of the *Trivia Game* on Side 1 of the tape. We really strained our brains for those questions, but now it is up to you. If you have a copy of *Trivial Pursuit* (or some such game), you could enter some selected questions and play with the computer keeping score. If not, there are some other uses for the data base entry program and game program.

If you have students who need to drill on facts (answers no longer than two words), they could have the questions entered (by you?), and then practice with the computer's assistance. If you are studying for a professional exam, the same sort of assistance is available for you.

Whether you use these two programs for entertainment or education, it is our hope that you will gain from their purchase. Also, a thorough examination of the code structure and subroutines will be of great assistance in any other data base type programs which you might wish to write for your TI- 99/4A.

Watch for other entertaining and instructional programs from us for your TI-99/4A. TI may have given up on you, but we haven't.

Appendix A
The Cassette Data Base

```
10 REM TRIVIA DB-CASSETTE VERSION
20 OPTION BASE 1 :: DIM QA$(30)
30 CALL CHAR(128,"FF00000000000000"):: CALL  CHAR
   (129,"0000 7E7E7E7E0000"):: CALL CHAR
   (136,"FFFFFFFFFFFFFFFF")
40 CALL COLOR(14,13,4):: BLK$=RPT$(" ",78):: FALSE=0
   ::TRUE=NOT FALSE :: GOTO 670
50 Q$=SEG$(Q$&BLK$,1,78):: ANS$=SEG$(ANS$&BLK$,1,26)::
   QA$(ARRNUM)=Q$&ANS$ :: RETURN
60 ANS$=A$ :: ERR=FALSE :: CALL PARSEANS(ANS$,ERR)::
   RETURN
70 CALL CLEAR :: DE=FALSE
80 ARRNUM=ARRNUM+1 :: IF ARRNUM>30 THEN ARRNUM=30 ::
   CALL FULL :: RETURN
90 TT$="**  ADDING QUESTIONS  **" :: CALL  SCRFORM
   (ARRNUM,TT$):: CALL ADDCMDS
100 Q$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE :: CALL
   GETCHAR(7,2,3,Q$,FLAG,SV,EX)::IF FLAG=1 THEN CALL
   ERASEQUEST :: GOTO 100
110 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE IF
   FLAG=2 AND Q$="" THEN 100
120 Q$=SEG$(Q$&BLK$,1,78):: CALL WRAP(Q$,BLK$)
130 A$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE :: CALL
   GETCHAR(17,2,1,A$,FLAG,SV,EX)
140 IF FLAG=1 THEN CALL ERASEQUEST :: CALL ERASEANS ::
   GOTO 100
150 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE
   GOSUB 60 :: IF ERR THEN CALL ERRMSG :: GOTO 130
160 IF FLAG=2 THEN GOSUB 50 :: CHNGS=TRUE :: GOTO 80
170 CALL HCHAR(23,31,129):: CALL BEEP
180 CALL KEY(0,K,S):: IF S<=0 THEN 180
190 IF K=146 THEN GOSUB 230 :: CALL ADDCMDS :: GOTO
   170
200 IF K=147 THEN FLAG=2 :: GOTO 160
210 IF K=152 THEN ARRNUM=ARRNUM-1 :: RETURN ELSE CALL
   BEEP :: GOTO 180
220 Q$=SEG$(QA$(ARRNUM),1,78)::  ANS$=SEG$
   (QA$(ARRNUM),79,26)
230 BOTH=FALSE :: DISPLAY AT(22,2)BEEP:"REDO:" ::
   DISPLAY AT(23,1)SIZE(28):"  Q- QUESTION, A -
   ANSWER"
240 DISPLAY AT(24,1)SIZE(28):"  B - BOTH, X - EXIT"
250 CALL KEY(0,K,S):: IF S<=0 THEN 250
260 IF K<>81 AND K<>113 AND K<>65 AND K<>97 AND K<>66
   AND K<>98 AND K<>88 AND K<>120 THEN 250
270 IF K<>88 AND K<>120 THEN CALL PRRNT :: CHNGS=TRUE
```

```
280 IF K=81 OR K=113 THEN CALL ERASEQUEST :: GOTO 300
    ELSE IF K=65 OR K=97 THEN CALL ERASEANS :: GOTO
    320
290 IF K=66 OR K=98 THEN BOTH=TRUE :: CALL ERASEQUEST
    :: CALL ERASEANS :: GOTO 300 ELSE RETURN
300 SV=FALSE :: EX=FALSE :: Q$="" :: FLAG=0 :: CALL
    GETCHAR(7,2,3,Q$,FLAG,SV,EX):: IF FLAG=1 THEN CALL
    ERASEQUEST :: GOTO 300
310 Q$=SEG$(Q$&BLK$,1,78):: CALL WRAP(Q$,BLK$):: IF
    NOT BOTH THEN GOSUB 50 :: GOTO 220
320 SV=FALSE :: EX=FALSE :: A$="" :: FLAG=0 :: CALL
    GETCHAR(17,2,1,A$,FLAG,SV,EX)
330 IF FLAG=1 THEN CALL ERASEANS :: GOTO 320 ELSE
    GOSUB 60 :: IF ERR THEN CALL ERRMSG :: GOTO 320
    ELSE GOSUB 50 :: GOTO 220
340 IF ARRNUM>0 THEN 370
350 CALL CLEAR :: DISPLAY AT(6,2)BEEP:"THERE ARE NO
    QUESTIONS"
360 DISPLAY AT(8,2):"AND ANSWERS IN MEMORY TO" ::
    DISPLAY AT(10,2):"DISPLAY/EDIT." :: CALL ENTER ::
    RETURN
370 ARRPOS=1 :: DE=TRUE
380 TT$="* DISPLAY/EDIT QUESTION *" :: CALL
    SCRFORM(ARRPOS,TT$)
390 DISPLAY AT(7,2):SEG$(QA$(ARRPOS),1,26):: DISPLAY
    AT(9,2):SEG$(QA$(ARRPOS),27,26)
400 DISPLAY AT(11,2):SEG$(QA$(ARRPOS),53,26):: DISPLAY
    AT(17,2):SEG$(QA$(ARRPOS),79,26)
410 CALL EDITCMDS
420 CALL KEY(5,K,S):: IF S<=0 THEN 420
430 IF K=132 THEN 530 ELSE IF K=142 THEN
    ARRPOS=ARRPOS+1 :: GOTO 470
440 IF K=144 THEN ARRPOS=ARRPOS-1 :: GOTO 500
450 IF K=146 THEN TOT=ARRNUM :: ARRNUM=ARRPOS :: GOSUB
    220 :: GOSUB 50 :: ARRNUM=TOT :: GOTO 380
460 IF K=152 THEN RETURN ELSE CALL BEEP :: GOTO 420
470 IF ARRPOS<=ARRNUM THEN CALL NUMDISP(ARRPOS):: GOTO
    390
480 DISPLAY AT(23,1)BEEP:"THERE ARE NO MORE
    QUESTIONS!":: DISPLAY AT(24,1)SIZE(28)
490 CALL DELAY(600):: CALL EDITCMDS :: CALL BEEP ::
    ARRPOS=ARRPOS-1 :: GOTO 420
500 IF ARRPOS>=1 THEN CALL NUMDISP(ARRPOS):: GOTO 390
510 DISPLAY AT(23,1)BEEP:"CAN'T GO BELOW RECORD #1!"
    ::DISPLAY AT(24,1)SIZE(28)
520 CALL DELAY(600):: CALL EDITCMDS :: CALL BEEP ::
    ARRPOS=ARRPOS+1 :: GOTO 420
530 DISPLAY AT(23,1)BEEP:"DELETE THIS RECORD? (Y/N)"
    ::DISPLAY AT(24,1)SIZE(28)
540 CALL KEY(0,K,S):: IF S<=0 THEN 540
550 CALL HCHAR(23,30,K):: IF K<>78 AND K<>89 AND
    K<>110 AND K<>121 THEN CALL BEEP :: GOTO 540
560 IF K=78 OR K=110 THEN 410 ELSE IF ARRNUM-1=0 THEN
    QA$(ARRPOS)="" :: ARRNUM=0:: GOTO 350
570 IF ARRPOS=ARRNUM THEN QA$(ARRNUM)="" ::
    ARRPOS=ARRPOS-1 ELSE QA$(ARRPOS)=QA$(ARRNUM)
```

```
580 ARRNUM=ARRNUM-1 :: CHNGS=TRUE :: IF ARRNUM<1 THEN
    350 ELSE CALL NUMDISP(ARRPOS):: GOTO 390
590 CALL SAVEMSG :: CALL CLEAR
600 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 128
610 PRINT #1:NUMQ :: FOR I=1 TO NUMQ :: PRINT
    #1:QA$(I):: NEXT I
620 CLOSE #1 :: RETURN
630 IF NOT CHNGS THEN IF SL=4 THEN CALL CLEAR :: END
    ELSE 670
640 NUMQ=ARRNUM :: YFLG=FALSE :: CALL SAVECHNGS(YFLG)
650 IF NOT YFLG THEN IF SL=4 THEN CALL CLEAR :: END
    ELSE 670
660 GOSUB 590 :: GOTO 670
670 CHNGS=FALSE :: NUMQ=0 :: CALL CLEAR :: CALL
    SCREEN(4):: DISPLAY AT(1,8)BEEP:"** TRIVIA DB **"
680 DISPLAY AT(5,2):"DO YOU WANT TO:" :: DISPLAY
    AT(8,3):"1. LOAD AN EXISTING FILE"
690 DISPLAY AT(10,3):"2. CREATE A NEW FILE" :: DISPLAY
    AT(13,3):"PLEASE SELECT (1-2)"
700 CALL KEY(0,K,S):: IF S<=0 THEN 700
710 DISPLAY AT(13,23):CHR$(K):: IF (K<49)OR(K>50)THEN
    DISPLAY AT(13,23)BEEP SIZE(1):: GOTO 700
720 IF K=50 THEN 790 ELSE DISPLAY AT(5,2)ERASE ALL
    BEEP:"PLEASE INSERT A CASSETTE"
730 DISPLAY AT(7,2):"WHERE YOUR QUESTIONS AND" ::
    DISPLAY AT(9,2):"ANSWERS HAVE BEEN SAVED."
740 DISPLAY AT(13,2):"THEN FOLLOW THE DIRECTIONS" ::
    DISPLAY AT(15,2):"AS THEY ARE PRESENTED ON"
750 DISPLAY AT(17,2):"THE SCREEN." :: CALL ENTER ::
    CALL CLEAR
760 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
770 INPUT #1:NUMQ :: FOR I=1 TO NUMQ :: INPUT
    #1:QA$(I):: NEXT I
780 CLOSE #1
790 ARRNUM=NUMQ
800 DISPLAY AT(4,3)ERASE ALL BEEP:"** TRIVIA DB MAIN
    MENU **" :: DISPLAY AT(8,6):"1. ADD QUESTIONS"
810 DISPLAY AT(10,6):"2. DISPLAY/EDIT" :: DISPLAY
    AT(11,9):"QUESTIONS" :: DISPLAY AT(13,6):"3.
    CHANGE FILES"
820 DISPLAY AT(15,6):"4. EXIT PROGRAM" :: DISPLAY
    AT(18,6):"PLEASE SELECT (1-4)"
830 CALL KEY(0,K,S):: IF S<=0 THEN 830
840 DISPLAY AT(18,26):CHR$(K):: IF (K>48)AND(K<53)THEN
    K=K-48 :: SL=K ELSE DISPLAY AT(18,26)BEEP
    SIZE(1):: GOTO 830
850 ON K GOSUB 70,340,630,630
860 GOTO 800
870 SUB SAVEMSG
880 CALL CLEAR :: DISPLAY AT(5,2)BEEP:"PLEASE INSERT A
    CASSETTE"
890 DISPLAY AT(7,2):"WHERE YOU WANT YOUR NEW" ::
    DISPLAY AT(9,2):"QUESTIONS AND ANSWERS TO"
900 DISPLAY AT(11,2):"BE SAVED." :: DISPLAY
    AT(14,2):"THEN FOLLOW THE DIRECTIONS" :: DISPLAY
    AT(16,2):"AS THEY ARE PRESENTED ON"
```

```
910 DISPLAY AT(18,2):"THE SCREEN." :: CALL ENTER
920 SUBEND
930 SUB SCRFORM(NUMREC,TT$)
940 CALL CLEAR :: CALL OUTLINE :: DISPLAY
    AT(1,3)BEEP:TT$
950 DISPLAY AT(5,2):"QUESTION#";NUMREC :: CALL
    HCHAR(8,4,128,26):: CALL HCHAR(10,4,128,26):: CALL
    HCHAR(12,4,128,26)
960 DISPLAY AT(15,2):"ANSWER FOR QUESTION#";NUMREC ::
    CALL HCHAR(18,4,128,26)
970 DISPLAY AT(19,6):"(USE ONLY 2 WORDS)"
980 SUBEND
990 SUB GETCHAR(STROW,STCOL,NROWS,ANS$,FLG,SV,EX)
1000 ROW=STROW :: COL=STCOL :: NR=1 :: ANS$=""
1010 CALL HCHAR(ROW,COL+2,129):: CALL KEY(5,K,S):: IF
    S=0 THEN 1010
1020 IF K=146 THEN FLG=1 :: GOTO 1200
1030 IF (K=147)AND SV THEN FLG=2 :: GOTO 1190 ELSE IF
    K=147 THEN 1010
1040 IF (K=152)AND EX THEN FLG=3 :: GOTO 1200 ELSE IF
    K=152 THEN 1010
1050 IF (K<>13)AND(K<>8)AND((K<32)OR(K>126))THEN CALL
    BEEP :: GOTO 1010
1060 IF K=13 THEN 1190 ELSE IF K=8 THEN 1090
1070 ANS$=ANS$&CHR$(K):: DISPLAY AT(ROW,COL):CHR$(K)::
    COL=COL+1 :: IF COL<=27 THEN 1010
1080 ROW=ROW+2 :: NR=NR+1 :: IF NR>NROWS THEN 1120
    ELSE COL=STCOL :: GOTO 1010
1090 COL=COL-1 :: IF (COL<=STCOL)AND(NR=1)THEN CALL
    HCHAR(ROW,COL+3,32):: CALL BEEP :: GOTO 1000
1100 IF COL<STCOL THEN CALL HCHAR(ROW,COL+3,32)::
    COL=27 :: NR=NR-1 :: ROW=ROW-2
1110 CALL HCHAR(ROW,COL+3,32)::  ANS$=SEG$
    (ANS$,1,LEN(ANS$)-1):: GOTO 1010
1120 ROW=ROW-2 :: NR=NR-1 :: CALL HCHAR(ROW,30,129)
1130 CALL KEY(0,K,S):: IF S<=0 THEN 1130
1140 IF K=146 THEN FLG=1 :: GOTO 1200
1150 IF (K=147)AND SV THEN FLG=2 :: GOTO 1190 ELSE IF
    K=147 THEN 1130
1160 IF (K=152)AND EX THEN FLG=3 :: GOTO 1200 ELSE IF
    K=152 THEN 1130
1170 IF (K<>13)AND(K<>8)THEN CALL BEEP :: GOTO 1130
1180 IF K=13 THEN 1190 ELSE 1090
1190 IF ANS$="" THEN CALL BEEP :: GOTO 1010 ELSE CALL
    HCHAR(ROW,COL+2,32)
1200 SUBEND
1210 SUB BEEP
1220 CALL SOUND(100,523,0)
1230 SUBEND
1240 SUB OUTLINE
1250 CALL HCHAR(3,3,136,29):: CALL  HCHAR
    (21,3,136,29)::CALL VCHAR(3,3,136,18)::
     CALL VCHAR(3,31,136,18)
1260 SUBEND
1270 SUB DELAY(TIME)
1280 FOR I=1 TO TIME :: NEXT I
```

```
1290 SUBEND
1300 SUB ENTER
1310 DISPLAY AT(22,2):"PRESS <ENTER> TO CONTINUE"
1320 CALL KEY(0,K,S):: IF S<=0 THEN 1320
1330 IF K<>13 THEN 1320
1340 SUBEND
1350 SUB PARSEANS(ANS$,ERR)
1360 WD$="" :: WRDS=0 :: FOR I=1 TO LEN(ANS$)
1370 IF ASC(SEG$(ANS$,I,1))=32 AND WD$="" THEN 1410
1380 IF SEG$(ANS$,I,1)=" " AND SEG$(ANS$,I+1,1)=" "
     THEN 1410
1390 IF ASC(SEG$(ANS$,I,1))=32 OR I=LEN(ANS$)THEN
     WRDS=WRDS+1 :: IF WRDS>2 THEN ERR=-1 :: GOTO 1480
1400 WD$=WD$&SEG$(ANS$,I,1)
1410 NEXT I :: T$="" :: FOR J=1 TO LEN(WD$)::
     A=ASC(SEG$(WD$,J,1))
1420 IF A>96 AND A<123 THEN A=A-32
1430 T$=T$&CHR$(A):: NEXT J
1440 FOR I=1 TO LEN(T$):: IF ASC(SEG$(T$,I,1))=32 AND
     I<>1 THEN 1460
1450 NEXT I :: GOTO 1480
1460 A$=SEG$(T$,1,I):: IF (A$="A " OR A$="THE " OR
     A$="AN ")AND I<>LEN(T$)THEN T$=SEG$(T$,I+1,LEN(T$)-
     I)
1470 ANS$=T$
1480 SUBEND
1490 SUB ERASEQUEST
1500 DISPLAY AT(7,2)SIZE(26):: DISPLAY  AT
     (9,2)SIZE(26):: DISPLAY AT(11,2)SIZE(26)
1510 SUBEND
1520 SUB ERASEANS
1530 DISPLAY AT(17,2)SIZE(26)
1540 SUBEND
1550 SUB FULL
1560 CALL CLEAR :: CALL BEEP :: CALL BEEP
1570 DISPLAY AT(12,2):"MEMORY IS FULL!!" :: CALL ENTER
1580 SUBEND
1590 SUB ADDCMDS
1600 DISPLAY AT(22,1)SIZE(28):: DISPLAY  AT
     (23,1)SIZE(28):"C/S-SAVE, C/R-REDO, C/X-EXIT"
1610 DISPLAY AT(24,1)SIZE(28)
1620 SUBEND
1630 SUB PRRNT
1640 DISPLAY AT(22,1)SIZE(28):: DISPLAY AT(23,1):"
     PRESS <ENTER> WHEN DONE!" ::DISPLAY AT
     (24,1)SIZE(28)
1650 CALL BEEP
1660 SUBEND
1670 SUB SAVECHNGS(YFLG)
1680 DISPLAY AT(10,3)ERASE ALL BEEP:"SAVE CHANGES?
     (Y/N)"
1690 CALL KEY(0,K,S):: IF S<=0 THEN 1690 ELSE CALL
     HCHAR(10,25,K)
1700 IF K<>78 AND K<>110 AND K<>89 AND K<>121 THEN
     CALL HCHAR(10,25,32):: GOTO 1690
1710 IF K=78 OR K=110 THEN YFLG=0 ELSE YFLG=-1
```

```
1720 SUBEND
1730 SUB ERRMSG
1740 DISPLAY AT(20,7)BEEP:"RE-ENTER ANSWER" :: CALL
    DELAY(400)
1750 DISPLAY AT(20,7)SIZE(17):: DISPLAY  AT
    (17,2)SIZE(26)
1760 SUBEND
1770 SUB NUMDISP(ARRCTR)
1780 DISPLAY AT(5,11)SIZE(4):ARRCTR :: DISPLAY  AT
    (15,22)SIZE(4):ARRCTR
1790 SUBEND
1800 SUB EDITCMDS
1810 DISPLAY AT(22,2):"C/R-REDO, C/N-NEXT, C/P-PREVC/D-
    DEL, C/X-EXIT"
1820 SUBEND
1830 SUB WRAP(Q$,BLK$)
1840 WL=0
1850 WL=WL+1 :: IF WL>2 THEN 1910
1860 IF ASC(SEG$(Q$,WL*26,1))=32 THEN 1850
1870 ST=WL*26-1 :: EN=(WL-1)*26+1 :: FOR I=ST TO EN
    STEP -1 :: IF ASC(SEG$(Q$,I,1))=32 THEN 1890
1880 NEXT I :: GOTO 1850
1890 T$=SEG$(Q$,I+1,WL*26-I)::  T2$=SEG$(SEG$(SEG$
    (Q$,1,I)&BLK$,1,WL*26)&T$&SEG$(Q$,WL*26+1,
    (WL+1)*26),1,78)
1900 Q$=T2$ :: GOTO 1850
1910 DISPLAY AT(7,2):SEG$(Q$,1,26):: DISPLAY  AT
    (9,2):SEG$(Q$,27,26)
1920 DISPLAY AT(11,2):SEG$(Q$,53,26)
1930 SUBEND
```

Variable Listing — Main Program

A$	Temporarily holds the answer before packing and concatenating with the question.
ANS$	Holds the answer. Maximum length is 26 characters.
ARRNUM	Holds the array number of the next location to store the question and answer.
ARRPOS	Holds the current location of the array QA$ for displaying the record to the screen. Used in Display/Edit Menu.
BLK$	Blank string the size of the maximum size field (78). Used for packing the question and answer to their maximum length.

BOTH	Boolean flag set to indicate if the user wants to REDO both the question and the answer. If true, the user wants to edit both parts of the record.
CHNGS	Boolean flag. Indicates if any changes have been made to the existing file in memory.
DE	Flag set to indicate whether the program execution came from the *D*isplay/*E*dit menu. If true, then it did. Make sure you return to Display/Edit menu.
ERR	Flag used to indicate if an error occurred within a procedure. This type of an error is not a system error but a user error (invalid response or illegal characters, etc.).
EX	Boolean flag. Indicates whether or not to allow the user to use the C/X key.
FALSE	Boolean flag set to 0. Indicates a false value.
FLAG	Flag set to indicate if one of the command keys (C/S, C/X, etc.) has been pressed. Returns a 1 if C/R is pressed, a 2 if C/S is pressed, and a 3 if C/X has been pressed.
I	Looping variable for FOR...NEXT statements.
K	Used in the CALL KEY routine to hold the ASCII value of the key that is pressed.
NUMQ	Number of questions currently loaded into memory or set to 0 initially for creating a new file.
QA$(30)	Holds the questions and answers in a concatenated string. Total string length is 104 characters, 78 for the question and 26 for the answer.
Q$	Holds the question. Maximum length is 78 characters.

S	Used in the CALL KEY routine to hold the status of the keyboard.
SL	Holds the selection number the user chose from the Main Menu.
SV	Boolean flag. Indicates whether or not to allow the user to press C/S while typing in the question or answer.
TOT	Temporarily holds the total number of questions and answers in memory.
TRUE	Boolean flag indicates that the result is indeed true. Usually a −1, but the TI-99/4A accepts any number other than 0 to be true.
TT$	Holds the title string to be printed at the top of each screen.
YFLG	Boolean flag to indicate whether the user responded with a YES or a NO.

Subroutines

Here are the subroutines that are used in the Database program with an explanation of what they do and the parameters needed to use them.

SUB SAVEMSG

This routine displays a message to the screen telling the user to insert a cassette on which he wants to save the file. Tells user to then follow the instructions as they appear on the screen.

SUB SCRFORM(NUMREC,TT$)

This routine displays the screen format for both the ADD QUESTION and the DISPLAY/EDIT QUESTION routines. The NUMREC parameter is the current record number to be displayed, and it is printed as the question number. The TT$ is the title of the screen. It is always printed at the top of the screen. TT$ is either ADDING QUESTION or DISPLAY/EDIT QUESTION. The screen is then displayed showing the border and the lines where the question and answer will be displayed.

SUB GETCHAR(STROW,STCOL,NROWS,ANS$,FLG,SV,EX)

This routine is the heart of adding or editing a question or answer. This routine gets one character at a time and displays it in the appropriate row and column. It uses the CALL KEY routine. This routine is called whenever a question or answer is needed. STROW is the starting row position for the character to be displayed. STCOL is the starting column position. NROWS is how many rows the answer can contain. (For the QUESTION line, the NROWS is 3 because the answer can be 78 characters long and there can only be 26 characters displayed per line.) ANS$ contains the concatenated string of all the characters that were typed by the user. It is the variable that the question and the answer is stored in before returning. FLG is a flag used to tell if one of the command keys has been pressed. (FLG returns a 1 if the C/R has been pressed. A 2 is returned if the C/S has been pressed, and a 3 is returned if the C/X has been pressed. The program can then act on whatever the flag has been set.

SUB BEEP

This subroutine produces a beeping sound to indicate that an invalid response has been entered.

SUB OUTLINE

This routine draws the outline border for the add and edit screens. Inside this screen is where the question and answer will be displayed.

SUB DELAY(TIME)

This routine is a pausing routine. It merely holds the screen the way it is for a certain time period. TIME is the length for how long to hold the screen. It uses a FOR...NEXT loop to stall the screen.

SUB ENTER

This routine displays this message PRESS <ENTER> TO CONTINUE and then waits for the **ENTER** key to be pressed before returning to the main program.

SUB PARSEANS(ANS$,ERR)

This routine parses through the answer to make sure there were no more than two words in the answer. It also checks to see if A, AN, and THE are used in conjunction with another word. If they were, the answer is stripped of them. ANS$ is the string

where the corrected (an answer without A, AN, or THE) answer is returned. ERR is a flag that indicates if the answer is more than two words long. If it is, this variable is set to true and the program can act on it when control returns to the main program.

SUB ERASEQUEST

This routine erases the question from the screen. Used whenever the question has to be redone.

SUB ERASEANS

This routine erases the answer from the screen. Used whenever the answer has to be redone.

SUB FULL

Displays the message MEMORY IS FULL!! and calls the routine to wait for the **ENTER** key to be pressed.

SUB ADDCMDS

This subroutine displays the ADD QUESTION command line at the bottom of the screen.

SUB PRRNT

This subroutine displays the message for the user to press **ENTER** when done.

SUB SAVECHNGS(YFLG)

Takes a flag parameter that can be set to true if the user presses the Y key or to false if the user presses the N key. Waits for a Y or an N key to be pressed.

SUB ERRMSG

Prints the error message "RE-ENTER ANSWER" and is used when the answer given is invalid.

SUB NUMDISP(ARRCTR)

This subroutine displays the number for the questions and answers on the screen. ARRCTR is the current number for the question that is being displayed or added.

SUB EDITCMDS

This subroutine displays the commands the user is able to use during the DISPLAY/EDIT screen.

SUB WRAP(Q$,BLK$)

This subroutine takes the question that the user has typed on the screen and reformats it to make it look nicer on the screen. Words are wrapped around to the next line. Q$ is the question that needs to be wrapped around on the screen. BLK$ is the blank string used for packing the question to its fullest length of 78 characters.

Variable Listing for SUBROUTINES

SUB SCRFORM(NUMREC,TT$)

NUMREC	Record number to print. Same as the question number. Shows what question the user is currently entering.
TT$	Same purpose, as TT$ is in the MAIN PROGRAM.

SUB GETCHAR(STROW,STCOL,NROWS,ANS$,FLG,SV,EX)

STROW	Starting row location to place the cursor and where to start printing characters as they are typed.
STCOL	Starting column location for placing the cursor and displaying the characters.
NROWS	The number of rows the cursor is allowed to travel. For the question, NROWS is set to 3; for the answer, it is set to 1.
ANS$	Holds the answer or the question as the user types in the characters.
FLG	Flag returns a 1,2, or 3 depending on which, if any, of the three command keys (C/S, C/R, C/X) was pressed.
SV	Tells subroutine whether or not to allow the user to press C/S as a valid key.
EX	Tells subroutine whether or not to accept C/X as a valid input.
ROW	Is used for keeping the cursor on the same line until it reaches the end of the line then move it down to the next line.
COL	Used for keeping the cursor, as well as the characters, placed one in front of the other.
NR	Counter to keep track of how many rows the cursor has traveled compared to how many it is allowed to move.

K & S	Same purpose here as in the MAIN PROGRAM.

SUB DELAY(TIME)

TIME	Factor for how long the FOR...NEXT loop should go for a delaying time period.
I	Same purpose as in the MAIN PROGRAM.

SUB ENTER

K & S	Same purpose as in the MAIN PROGRAM.

SUB PARSEANS(ANS$,ERR)

ANS$	Holds the answer that you want parsed. Parses out the words THE, A, and AN from the answer. Returns the correct answer in this variable.
ERR	Is set if the answer contains more than two words.
WD$	Temporarily holds the new answer after parsing.
WRDS	Counter used to count how many words are in your answer.
I & J	Looping variables.
T$	Temporarily holds the answer after it is converted to all uppercase letters.
A	Holds the ASCII value of each character in the answer while converting the character to uppercase.
A$	Holds the first word of the answer to see if it matches A, AN, or THE.

SUB SAVECHNGS(YFLG)

YFLG	Holds a true value if the user presses a Y; false if the user presses an N. Returns one of the two above to the MAIN PROGRAM for execution on the flag.
K & S	Same purpose as in the MAIN PROGRAM.

SUB NUMDISP(ARRCTR)

ARRCTR	The question number that is displayed on the screen. Indicates the current question.

SUB WRAP(Q$,BLK$)

Q$	Holds the question that is to be wrapped around the screen.
BLK$	Same purpose as in the MAIN PROGRAM.
WL	Indicates Which Line we are currently calculating for size.
ST	Starting character position within the string Q$.
EN	Ending position within the Question string.
I	Same as in MAIN PROGRAM.
T$	Temporarily holds the new reformatted question (1st part).
T2$	Temporarily holds the rest of the string in its newly reformatted state.

Appendix B
The Cassette Game

```
10 REM TRIVIA GAME - CASSETTE VERSION
20 CALL CLEAR :: CALL SCREEN(4):: OPTION BASE 1 :: DIM
   QA$(30),RQN(30),N(30),ND(8),NT(4)
30 BLK$=RPT$(" ",26):: FALSE=0 :: TRUE=NOT FALSE ::
   CALL CHAR(128,"FF00000000000000"):: CALL
   CHAR(129,"00007E7E7E7E0000")
40 CALL CHAR(136,"FFFFFFFFFFFFFFFF"):: CALL
   COLOR(14,13,4):: GOTO 200
50 DISPLAY AT(19,2):"PLAYER#";PLYR :: DISPLAY  AT
   (19,15):"NAME: ";SEG$(PN$(PLYR),1,10)
60 DISPLAY AT(21,2):"SCORE:";SC(PLYR):: DISPLAY  AT
   (21,15):"NC/NQ:";NC(PLYR);"/";NQ(PLYR):: RETURN
70 CALL SOUND(300,-1,0):: CALL SOUND(300,-1,0)::
   DISPLAY AT(1,10)ERASE ALL:"FINAL STATS"
80 DISPLAY AT(5,2):"NAME         SCORE   NC/NQ" :: CALL
   HCHAR(6,4,128,29)
90 IF NP=1 THEN DISPLAY AT(8,2):PN$(1):: DISPLAY  AT
   (8,14):SC(1):: DISPLAY AT(8,22):NC(1);"/";NQ(1)::
   GOTO 160
100 FOR I=1 TO NP :: PN(I)=I :: NEXT I :: FOR J=1 TO
   NP-1 :: J1=J
110 FOR K=J+1 TO NP :: IF SC(J1)<SC(K)THEN J1=K
120 NEXT K :: IF J1<>J THEN T=SC(J):: SC(J)=SC(J1)::
   SC(J1)=T :: T=PN(J):: PN(J)=PN(J1):: PN(J1)=T
130 NEXT J
140 FOR I=1 TO NP :: DISPLAY AT(6+I,2):PN$(PN(I))::
   DISPLAY AT(6+I,14):SC(I)
150 DISPLAY AT(6+I,22):NC(PN(I));"/";NQ(PN(I)):: NEXT
   I
160 IF QUIT THEN CALL DELAY(800):: GOTO 190
170 DISPLAY AT(22,2)BEEP:"ANOTHER GAME? (Y/N)" ::
   YFLG=FALSE :: CALL GETYN(YFLG)
180 IF YFLG THEN DISPLAY AT(10,2)ERASE ALL:"ONE MOMENT
   PLEASE..." :: RUN 10
190 DISPLAY AT(10,2)ERASE ALL:"HAVE A NICE DAY!" ::
   END
200 DISPLAY AT(6,4)BEEP:"WELCOME TO THE TI-99/4A" ::
   DISPLAY AT(10,6):"**  TRIVIA GAME  **" :: CALL
   ENTER
210 DISPLAY AT(8,2)ERASE ALL BEEP:"HOW MANY CASSETTE
   FILES" :: DISPLAY AT(10,2):"DO YOU HAVE? (1-10)"
220 ACCEPT  AT(10,21)VALIDATE(NUMERIC) SIZE(3)
   BEEP:NUMFILES :: IF NUMFILES<1 OR NUMFILES>10 THEN
   220
230 IF NUMFILES=1 THEN 380
240 DISPLAY AT(8,2)ERASE ALL BEEP:"PLEASE WAIT...I'M
   WORKING" :: RANDOMIZE :: FOR I=1 TO 30
```

```
250 WN=INT(RND*NUMFILES*30)+1 :: ERR=FALSE :: CALL
    CKNUM(WN,RQN(),ERR,30):: IF ERR THEN 250 ELSE
    RQN(I)=WN
260 NEXT I :: P=1 :: Q=30 :: T0=0
270 IF P>=Q THEN 360
280 V=RQN(P):: I=P :: J=Q+1
290 J=J-1 :: IF RQN(J)>V THEN 290
300 I=I+1 :: IF RQN(I)<V AND I<75 THEN 300
310 IF J>I THEN T=RQN(I):: RQN(I)=RQN(J):: RQN(J)=T ::
    GOTO 290
320 RQN(P)=RQN(J):: RQN(J)=V
330 IF (J-P)<(Q-J)THEN N(T0+1)=J+1 :: N(T0+2)=Q :: Q=J-
    1 :: GOTO 350
340 N(T0+1)=P :: N(T0+2)=J-1 :: P=J+1
350 T0=T0+2 :: GOTO 270
360 IF T0<>0 THEN Q=N(T0):: P=N(T0-1):: T0=T0-2 ::
    GOTO 270
370 FOR I=1 TO 30 :: N(I)=0 :: NEXT I
380 DISPLAY AT(2,2)ERASE ALL BEEP:" THIS IS GOING TO
    TAKE A" :: DISPLAY AT(4,2):"WHILE, FOR ME TO LOAD
    THE"
390 DISPLAY AT(6,2):"QUESTIONS AND ANSWERS FROM" ::
    DISPLAY AT(8,2):"YOUR CASSETTE."
400 DISPLAY AT(10,2):"  PLEASE BE PATIENT AS I" ::
    DISPLAY AT(12,2):"LOAD THE RANDOM QUESTIONS"
410 DISPLAY AT(14,2):"FROM YOUR CASSETTE FILES." ::
    CALL ENTER :: CN=1 :: CTR=1:: RECCTR=0
420 DISPLAY AT(20,2)ERASE ALL BEEP:"INSERT CASSETTE
    #";CN
430 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
440 INPUT #1:NUMQ :: FOR I=1 TO NUMQ :: INPUT
    #1:QA$(CTR)
450 IF NUMFILES=1 THEN RQN(CTR)=I :: CTR=CTR+1 :: GOTO
    470
460 IF I+RECCTR=N(CTR)THEN CTR=CTR+1
470 NEXT I :: RECCTR=RECCTR+NUMQ :: CN=CN+1 :: CLOSE
    #1:: IF CN<=NUMFILES THEN 420
480 DISPLAY AT(6,2)ERASE ALL BEEP:"PLEASE WAIT..." ::
    DISPLAY AT(10,2):"SCRAMBLING QUESTIONS"
490 RANDOMIZE :: NUMQ=RECCTR :: IF NUMQ>30 THEN
    NUMQ=30
500 NUMDBL=INT(NUMQ*.25)+1 :: NUMTPL=INT(NUMQ*.10)+1
510 FOR I=1 TO NUMQ
520 WN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(WN,N(),ERR,NUMQ)
530 IF ERR THEN 520 ELSE N(I)=WN
540 NEXT I :: FOR I=1 TO NUMDBL
550 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(QN,ND(),ERR,8)
560 IF ERR THEN 550 ELSE ND(I)=QN
570 NEXT I :: FOR I=1 TO NUMTPL
580 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(QN,NT(),ERR,4):: IF ERR THEN 580
590 CALL CKNUM(QN,ND(),ERR,8):: IF ERR THEN 580
600 NT(I)=QN :: NEXT I
```

```
610 DISPLAY AT(10,2)ERASE ALL BEEP:"NUMBER OF PLAYERS
    (1-4)"
620 CALL KEY(0,K,S):: IF S<=0 THEN 620
630 CALL HCHAR(10,28,K):: IF K<49 OR K>52 THEN CALL
    BEEP :: CALL HCHAR(10,28,32):: GOTO 620
640 NP=K-48 :: FOR I=1 TO NP :: DISPLAY AT(10,2)ERASE
    ALL:"PLAYER#";I;"NAME:"
650 ACCEPT AT(10,18)VALIDATE(UALPHA)BEEP
    SIZE(11):PN$(I):: IF PN$(1)="" THEN 650
660 NEXT I :: DISPLAY AT(1,5)ERASE ALL BEEP:"**
    TRIVIA GAME  **" :: CALL HCHAR(2,3,136,29)
670 CALL HCHAR(18,3,136,29):: CALL VCHAR(2,3,136,16)::
    CALL VCHAR(2,31,136,16):: CALL VCHAR(18,3,136,6)
680 CALL VCHAR(18,31,136,6):: CALL HCHAR(24,3,136,29)
690 DISPLAY AT(17,6)SIZE(20):"(USE ONLY 2 WORDS)" ::
    DISPLAY AT(5,2):"QUESTION #"
700 DISPLAY AT(14,2):"ANSWER:" :: CALL
    HCHAR(16,4,128,26)
710 DISPLAY AT(23,6):"CTRL-Q TO QUIT" :: DBL=FALSE ::
    TPL=FALSE
720 PLYR=1 :: ARRLOC=1 :: FOR I=1 TO NP :: NQ(I)=0 ::
    NC(I)=0 :: SC(I)=0 :: NEXT I
730 TIM=0 :: CALL TIMER(TIM):: GOSUB 50 :: DISPLAY
    AT(5,13)SIZE(2):ARRLOC
740 DISPLAY AT(7,2):SEG$(QA$(N(ARRLOC)),1,26)::
    DISPLAY AT(9,2):SEG$(QA$(N(ARRLOC)),27,26)
750 DISPLAY AT(11,2):SEG$(QA$(N(ARRLOC)),53,26)::
    STPLYR=PLYR
760 FOR I=1 TO NUMDBL :: IF ND(I)=N(ARRLOC)THEN
    DBL=TRUE :: GOTO 790
770 NEXT I :: DBL=FALSE :: FOR I=1 TO NUMTPL :: IF
    NT(I)=N(ARRLOC)THEN TPL=TRUE :: GOTO 790
780 NEXT I :: TPL=FALSE
790 DISPLAY AT(15,2)SIZE(26):: DISPLAY
    AT(12,9)SIZE(15)
800 IF DBL THEN CALL BEEP :: DISPLAY AT(12,10):"DOUBLE
    VALUE" :: CALL BEEP
810 IF TPL THEN CALL BEEP :: DISPLAY AT(12,10):"TRIPLE
    VALUE" :: CALL BEEP
820 TU=FALSE :: ANS$="" :: QUIT=FALSE :: CALL
    GETANS(ANS$,15,2,26,QUIT,TIM,TU)::POINTS=25-TIM ::
    IF QUIT THEN 940
830 IF TU THEN 860 ELSE IF DBL THEN POINTS=POINTS*2
840 IF TPL THEN POINTS=POINTS*3
850 CALL PARSEANS(ANS$):: ANS$=SEG$(ANS$&BLK$,1,26)::
    CA$=SEG$(QA$(N(ARRLOC)),79,26):: CALL
    CONVERTUPPER(CA$):: IF CA$=ANS$ THEN 910
860 CALL SOUND(700,-3,0):: CALL DELAY(200)::
    NQ(PLYR)=NQ(PLYR)+1
870 PLYR=PLYR+1 :: IF PLYR>NP THEN PLYR=1
880 IF PLYR<>STPLYR THEN TIM=13 :: CALL TIMER(TIM)::
    GOSUB 50 :: GOTO 790
890 DISPLAY AT(14,2)BEEP:"THE CORRECT ANSWER IS:" ::
    DISPLAY AT(15,2):SEG$(QA$(N(ARRLOC)),79,26)
900 CALL DELAY(500):: DISPLAY AT  (14,2)SIZE(26):
    "ANSWER:" :: DISPLAY AT(15,2)SIZE(26):: GOTO 920
```

```
910 CALL SOUND(700,-1,0):: SC(PLYR)=SC(PLYR)+POINTS ::
    NC(PLYR)=NC(PLYR)+1 :: NQ(PLYR)=NQ(PLYR)+1 ::
    GOSUB 50  :: CALL DELAY(300)
920 ARRLOC=ARRLOC+1 :: IF ARRLOC>NUMQ THEN CALL
    DELAY(200):: GOTO 70 ELSE PLYR=STPLYR+1 :: IF
    PLYR>NP THEN PLYR=1
930 DBL=FALSE :: TPL=FALSE :: GOTO 730
940 DISPLAY AT(23,6)SIZE(20)BEEP:"ARE YOU SURE? (Y/N)"
    :: YFLG=FALSE :: CALL GETYN(YFLG)
950 IF YFLG THEN 70 ELSE DISPLAY
    AT(23,6)SIZE(20):"CTRL-Q TO QUIT" :: GOTO 820
960 SUB CONVERTUPPER(WD$)
970 T$="" :: FOR J=1 TO LEN(WD$):: A=ASC(SEG$
    (WD$,J,1)):: IF A>96 AND A<123 THEN A=A-32
980 T$=T$&CHR$(A):: NEXT J :: WD$=T$
990 SUBEND
1000 SUB GETANS(ANS$,STROW,STCOL,LN,QUIT,TIMR,TU)
1010 TT=0
1020 ROW=STROW :: COL=STCOL :: ANS$=""
1030 CALL HCHAR(ROW,COL+2,129):: CALL KEY(5,K,S)
1040 IF S=0 AND TT>=15 THEN TT=0 :: TIMR=TIMR+1 ::
    CALL TIMER(TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO
    1240
1050 IF S=0 THEN TT=TT+1 :: GOTO 1030
1060 IF K=145 THEN QUIT=-1 :: GOTO 1210
1070 IF (K<>13)AND(K<>8)AND((K<32)OR(K>126))THEN CALL
    BEEP :: GOTO 1030
1080 TT=TT+1 :: IF TT>=15 THEN TT=0 :: TIMR=TIMR+1 ::
    CALL TIMER(TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO
    1240
1090 IF K=13 THEN 1200 ELSE IF K=8 THEN 1110
1100 ANS$=ANS$&CHR$(K):: DISPLAY AT(ROW,COL):CHR$(K)::
    COL=COL+1 :: IF COL<STCOL+LN THEN 1030 ELSE CALL
    BEEP :: GOTO 1130
1110 COL=COL-1 :: IF (COL<=STCOL)THEN CALL
    HCHAR(ROW,COL+3,32):: CALL BEEP :: GOTO 1020
1120 ANS$=SEG$(ANS$,1,LEN(ANS$)-1):: CALL
    HCHAR(ROW,COL+3,32):: GOTO 1030
1130 CALL HCHAR(ROW,COL+2,129):: CALL KEY(0,K,S)
1140 IF S<=0 THEN TT=TT+1
1150 IF TT>=15 THEN TT=0 :: TIMR=TIMR+1 :: CALL
    TIMER(TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO 1240
1160 IF S<=0 THEN TT=TT+1 :: GOTO 1130
1170 IF K=145 THEN QUIT=-1 :: GOTO 1210
1180 IF (K<>13)AND(K<>8)THEN CALL BEEP :: GOTO 1130
1190 IF K=13 THEN 1200 ELSE 1110
1200 IF ANS$="" THEN CALL BEEP :: GOTO 1030 ELSE CALL
    HCHAR(ROW,COL+2,32)
1210 SUBEND
1220 SUB BEEP
1230 CALL SOUND(100,523,0)
1240 SUBEND
1250 SUB PARSEANS(ANS$)
1260 WD$="" :: FOR I=1 TO LEN(ANS$)
1270 IF ASC(SEG$(ANS$,I,1))=32 AND WD$="" THEN 1300
```

```
128Ø IF SEG$(ANS$,I,1)=" " AND SEG$(ANS$,I+1,1)=" "
    THEN 13ØØ
129Ø WD$=WD$&SEG$(ANS$,I,1)
13ØØ NEXT I :: CALL CONVERTUPPER(WD$)
131Ø FOR I=1 TO LEN(WD$):: IF ASC(SEG$(WD$,I,1))=32
    AND I<>1 THEN 133Ø
132Ø NEXT I :: GOTO 135Ø
133Ø A$=SEG$(WD$,1,I):: IF (A$="A " OR A$="THE " OR
    A$="AN ")AND I<>LEN(WD$)THEN WD$=SEG$
    (WD$,I+1,LEN(WD$)-I)
134Ø ANS$=WD$
135Ø SUBEND
136Ø SUB TIMER(SEC)
137Ø DISPLAY AT(3,23)SIZE(4):25-SEC
138Ø SUBEND
139Ø SUB GETYN(YFLG)
14ØØ CALL KEY(Ø,K,S):: IF S<=Ø THEN 14ØØ
141Ø IF K<>78 AND K<>11Ø AND K<>89 AND K<>121 THEN
    14ØØ
142Ø IF K=78 OR K=11Ø THEN YFLG=Ø ELSE YFLG=-1
143Ø SUBEND
144Ø SUB ENTER
145Ø DISPLAY AT(22,1)SIZE(28):"PRESS <ENTER> TO
    CONTINUE"
146Ø CALL KEY(Ø,K,S):: IF S<=Ø THEN 146Ø ELSE IF K<>13
    THEN 146Ø
147Ø SUBEND
148Ø SUB CKNUM(NUMB,T(),ERR,MAXLIM)
149Ø FOR I=1 TO MAXLIM :: IF NUMB=T(I)THEN ERR=-1
15ØØ NEXT I
151Ø SUBEND
152Ø SUB DELAY(DURAT)
153Ø FOR K=1 TO DURAT :: NEXT K
154Ø SUBEND
```

Variable Listing — Main Program

ANS$	Holds the answer the player has typed in. Maximum length is 26 characters.
ARRLOC	Holds the array number of the question that is displayed.
BLK$	Blank string the size of the maximum size field (26). Used for packing the answer to its maximum length.
CA$	Holds the correct answer. Used for comparing the two answers for equality.
CN	Starting counter for looping through the number of cassette files the user has.

CTR	Counter for placing the correct random record into the array in a sequential manner.
DBL	Boolean flag to indicate if the current question is to be double value or not. Used for printing the DOUBLE VALUE message.
ERR	Flag used to indicate if an error occurred within a procedure.
FALSE	Boolean flag set to Ø. Indicates a false value.
I & J	Looping variable for FOR...NEXT statements.
K	Used in the CALL KEY routine to hold the ASCII value of the key that is pressed. K also used for looping.
N(30)	Holds the random generated sequence of numbers in which to display the questions in a random format.
NC()	Holds the player's number of correct questions that (s)he answered.
ND(8)	Holds the random generated numbers that are to be assigned DOUBLE point values.
NP	The total number of players playing the game, from one to four.
NQ()	Holds the number of questions that each player was asked.
NT(4)	Holds the random generated numbers that are to be assigned TRIPLE point value.
NUMFILES	Holds the total number of cassette files to load from. Maximum is 10.
NUMDBL	Holds the number of DOUBLE point questions this game is supposed to have.

NUMQ	Number of questions currently loaded into memory from the cassette file.
NUMTPL	Holds how many questions are to be TRIPLE pointers.
P	Holds the current pointer for the lowest partition using Quicksort routine.
PLYR	Holds the current player number to display on the screen.
PN$()	Holds the players' names. Holds up to 10 characters, but only six are displayed on the game screen. The whole name is displayed at the end of the game in the statistic screen.
Q	Holds the maximum array location to sort.
QA$(30)	Holds the questions and answers in a concatenated string. Total string length is 104 characters, 78 for the question and 26 for the answer.
QN	Holds the random question number for selecting the double and triple value questions without obtaining duplicate numbers.
QUIT	Holds the boolean flag for quitting the game early.
RECCTR	Holds the number of records loaded from each cassette file.
RQN(30)	Holds the Random Question Numbers to load in memory.
S	Used in the CALL KEY routine to hold the status of the keyboard.
SC()	Holds the player's score.

STPLYR	Holds the player number of who started first. This keeps the players going in the same order no matter who answers the questions from a missed question.
T	Used for holding the number that needs switching.
T0	Holds the stack pointer for Quicksorting.
TIM	Holds the timer value. Starts at 0 and is incremented until either the player answers the questions or it reaches the maximum point limit.
TPL	Holds the TRIPLE value flag set for printing the TRIPLE VALUE message or not.
TRUE	Boolean flag indicates that the result is indeed true. Usually a −1, but the TI-99/4A accepts any number other than 0 to be true.
TU	Boolean flag set only if the timer runs out of time. Indicates that Time is Up.
V	Temporarily holds a number to sort on.
WN	Holds a random number to select from the file.
YFLG	Boolean flag to indicate whether the user responded with a YES or a NO.

Subroutines for the Cassette Trivia Game

Here are the subroutines that are used in the Trivia Game program with an explanation of what they do and the parameters needed to use them.

SUB CONVERTUPPER(WD$)

This routine takes a string, WD$, and converts all the characters in the string to uppercase characters before comparing the answer the player typed to the correct answer.

SUB GETANS(ANS$,STROW,STCOL,LN,QUIT,TIMR,TU)

This routine is the heart of the answer routine. This routine gets one character at a time and displays it in the appropriate row and column. It uses the CALL KEY routine. This routine is called when a player is typing in the answer for the question. ANS$ is where the answer (s)he types is stored. STROW is the starting row position for the character to be displayed. STCOL is the starting column position. LN is the length of the answer (26 is maximum). QUIT is the quit flag for setting if the players wishes to exit the game early. If they do, then this flag gets set to true. TIMR is the timer passed to the subroutine so the routine can keep decreasing the time (point value). TU is the time up flag. It is set if the TIMR goes to the number equal to the starting value of the question.

SUB BEEP

This subroutine produces a beeping sound to indicate that an invalid response has been entered.

SUB DELAY(DURAT)

This routine is a pausing routine. It merely holds the screen the way it is for a certain time period. DURAT is the length for how long to hold the screen. It uses a FOR...NEXT loop to stall the screen.

SUB ENTER

This routine displays this message PRESS <ENTER> TO CONTINUE and then waits for the **ENTER** key to be pressed before returning to the main program.

SUB PARSEANS(ANS$)

This routine parses through the answer to make sure there were no more then two words in the answer. It also checks to see if A, AN, and THE are used in conjunction with another word. If they were, the answer is stripped of them. ANS$ is the string where the corrected (an answer without A, AN, or THE) answer is returned.

SUB TIMER(SEC)

This routine takes the time and displays the correct point value in the upper-right hand corner of the screen. SEC is the amount of time that has elapsed.

SUB GETYN(YFLG)

This routine waits for the user to answer a Y/N question with a Y or a N. If the user presses Y, the YFLG is set to TRUE; otherwise, it is set to false.

SUB CKNUM(NUMB,T(),ERR,MAXLIM)

This routine checks the array T() which is passed to it to make sure that the NUMB is not already in the array. If it is, ERR is set to false. MAXLIM is the maximum array dimension for the array T()

Variable Listing for SUBROUTINES

SUB CONVERTUPPER(WD$)

WD$	The string that needs to be converted to uppercase characters.
T$	Temporarily holds the new characters as they are converted.
J	Same purpose as in MAIN PROGRAM.
A	Holds the ASCII value for one character at a time.

GETANS(ANS$,STROW,STCOL,LN,QUIT,TIMR,TU)

ANS$	Holds the answer as the user types.
STROW	Starting row location to place the cursor and where to start printing characters as they are typed.
STCOL	Starting column location for placing the cursor and displaying the characters.

LN	Holds the length of the answer. (26 is the maximum amount.)
QUIT	Is the quit flag. Gets set if the user presses CTRL-Q.
TIMR	Holds the timer value as it clicks away.
TU	Flag that indicates if time ran out on the player before he could answer the question.
TT	Runs the timer from within the subroutine.
ROW	Holds the starting row location for displaying the printed characters.
COL	Holds the starting column number for displaying the characters side by side as they are typed.
K & S	Same purpose here as in the MAIN PROGRAM.

SUB DELAY(DURAT)

DURAT	Factor for how long the FOR...NEXT loop should go for a delaying time period.
K	Looping variable.

SUB ENTER

K & S	Same purpose as in the MAIN PROGRAM.

SUB PARSEANS(ANS$)

ANS$	Holds the answer that you want parsed. Parses out the words THE, A, and AN from the answer. Returns the correct answer in this variable.
WD$	Temporarily holds the new answer after parsing.
WRDS	Counter used to count how many words are in your answer.
I & J	Looping variables.

A$	Holds the first word of the answer to see if it matches A, AN, or THE.

SUB GETYN(YFLG)

YFLG	Holds a true value if the user presses a Y; false if the user presses an N. Returns one of the two above to the MAIN PROGRAM for execution on the flag.
K & S	Same purpose as in the MAIN PROGRAM.

SUB TIMER(SEC)

SEC	How many points to decrease the starting value by.

SUB CKNUM(NUMB,T(),ERR,MAXLIM)

NUMB	Number to check for duplication in selection random numbers.
T()	The array to compare the NUMB to, to see if any duplication is found.
ERR	Set to true if the NUMB already exists in the array T().
MAXLIM	Maximum amount of numbers to check in the array.
I	Looping variable.

Appendix C
The Diskette Data Base

```
10 REM
20 REM TRIVIA DB-DISKETTE VERSION
30 REM
40 CALL INIT :: CALL LOAD(-31806,16)
50 CALL CHAR(128,"FF00000000000000")::
    CALL CHAR(129,"00007E7E7E7E0000"):: CALL CHAR
    (136,"FFFFFFFFFFFFFFFF")
60 CALL COLOR(14,13,4):: BLK$=RPT$(" ",78)
70 FALSE=0 :: TRUE=NOT FALSE :: GOTO 1480
80 REM
90 REM STORE QUESTION AND ANSWER
100 REM
110 Q$=SEG$(Q$&BLK$,1,78)
120 ANS$=SEG$(ANS$&BLK$,1,26)
130 QA$=Q$&ANS$ :: RETURN
140 REM
150 REM PARSE ANSWER
160 REM
170 ANS$=A$ :: ERR=FALSE :: CALL PARSEANS(ANS$,ERR)::
    RETURN
180 REM
190 REM  ADD RECORDS
200 REM
210 CALL CLEAR :: DE=FALSE
220 ARRNUM=ARRNUM+1 :: IF ARRNUM>710 THEN CALL FULL ::
    RETURN
230 TT$="**  ADDING QUESTIONS  **"
240 CALL SCRFORM(ARRNUM,TT$):: CALL ADDCMDS
250 Q$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE
260 CALL GETCHAR(7,2,3,Q$,FLAG,SV,EX)
270 IF FLAG=1 THEN CALL ERASEQUEST :: GOTO 250
280 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN
290 IF FLAG=2 AND Q$="" THEN 260
300 Q$=SEG$(Q$&BLK$,1,78):: CALL WRAP(Q$,BLK$)
310 A$="" :: FLAG=0 :: SV=TRUE :: EX=TRUE :: CALL
    GETCHAR(17,2,1,A$,FLAG,SV,EX)
320 IF FLAG=1 THEN CALL ERASEQUEST :: CALL ERASEANS ::
    GOTO 250
330 IF FLAG=3 THEN ARRNUM=ARRNUM-1 :: RETURN
340 GOSUB 170 :: IF ERR THEN CALL ERRMSG :: GOTO 310
350 IF FLAG=2 THEN GOSUB 110 :: PRINT #1,REC
    ARRNUM:QA$:: GOTO 220
360 CALL HCHAR(23,31,129):: CALL BEEP
370 CALL KEY(0,K,S):: IF S<=0 THEN 370
380 IF K=146 THEN GOSUB 450 :: CALL ADDCMDS :: GOTO
    360
390 IF K=147 THEN FLAG=2 :: GOTO 350
```

```
400 IF K=152 THEN ARRNUM=ARRNUM-1 :: RETURN
410 CALL BEEP :: GOTO 370
420 REM
430 REM  RE-DO ROUTINE
440 REM
450 BOTH=FALSE :: DISPLAY AT(22,2)BEEP:"REDO:" ::
    DISPLAY AT(23,1)SIZE(28):"  Q- QUESTION, A
    ANSWER"
460 CALL HCHAR(23,31,32)
470 DISPLAY AT(24,1)SIZE(28):"  B - BOTH, X - EXIT"
480 CALL KEY(0,K,S):: IF S<=0 THEN 480
490 IF K<>81 AND K<>113 AND K<>65 AND K<>97 AND K<>66
    AND K<>98 AND K<>88 AND K<>120 THEN 480
500 IF K<>88 AND K<>120 THEN CALL PRRNT
510 IF K=81 OR K=113 THEN CALL ERASEQUEST :: GOTO 590
    ELSE IF K=65 OR K=97 THEN CALL ERASEANS :: GOTO
    680
520 IF K=66 OR K=98 THEN BOTH=TRUE :: CALL ERASEQUEST
    :: CALL ERASEANS :: GOTO 590
530 IF NOT DE THEN RETURN
540 IF NOT CHNGS THEN RETURN
550 PRINT #1,REC ARRNUM:QA$ :: RETURN
560 REM
570 REM  ENTER THE QUESTION
580 REM
590 SV=FALSE :: EX=FALSE
600 Q$="" :: FLAG=0 :: CALL GETCHAR(7,2,3,Q$,FLAG,SV,
    EX)
610 IF FLAG=1 THEN CALL ERASEQUEST :: GOTO 590 ELSE
    ANS$=SEG$(QA$,79,26)
620 Q$=SEG$(Q$&BLK$,1,78):: CALL WRAP(Q$,BLK$)::
    QA$=Q$&ANS$ :: CHNGS=TRUE
630 IF NOT BOTH THEN 450
640 QA$=SEG$(Q$&BLK$,1,78)
650 REM
660 REM  ENTER THE ANSWER
670 REM
680 SV=FALSE :: EX=FALSE :: A$="" :: FLAG=0 :: CALL
    GETCHAR(17,2,1,A$,FLAG,SV,EX)
690 IF FLAG=1 THEN CALL ERASEANS :: GOTO 680 ELSE
    Q$=SEG$(QA$,1,78):: GOSUB 170:: IF ERR THEN CALL
    ERRMSG :: GOTO 680
700 CHNGS=TRUE :: GOSUB 110 :: GOTO 450
710 REM
720 REM  DISPLAY/EDIT/DELETE QUESTIONS
730 REM
740 IF ARRNUM>0 THEN 780
750 CALL CLEAR :: DISPLAY AT(6,2)BEEP:"THERE ARE NO
    QUESTIONS"
760 DISPLAY AT(8,2):"AND ANSWERS IN MEMORY TO" ::
    DISPLAY AT(10,2):"DISPLAY/EDIT."
770 CALL ENTER :: RETURN
780 ARRPOS=1 :: DE=TRUE
790 TT$="* DISPLAY/EDIT QUESTION *" :: CALL
    SCRFORM(ARRPOS,TT$)
800 INPUT #1,REC ARRPOS:QA$
```

```
810 DISPLAY AT(7,2):SEG$(QA$,1,26):: DISPLAY AT
    (9,2):SEG$(QA$,27,26)
820 DISPLAY AT(11,2):SEG$(QA$,53,26):: DISPLAY AT
    (17,2):SEG$(QA$,79,26)
830 CALL EDITCMDS
840 CALL KEY(5,K,S):: IF S<=0 THEN 840
850 IF K=132 THEN 1070
860 IF K=140 THEN 1170
870 IF K=142 THEN ARRPOS=ARRPOS+1 :: GOTO 920
880 IF K=144 THEN ARRPOS=ARRPOS-1 :: GOTO 1000
890 IF K=146 THEN CHNGS=FALSE :: TOT=ARRNUM ::
    ARRNUM=ARRPOS :: GOSUB 450 :: GOSUB 110 ::
    ARRNUM=TOT :: GOTO 790
 900 IF K=152 THEN RETURN ELSE CALL BEEP :: GOTO 840
910 REM  GET NEXT RECORD
920 IF ARRPOS<=ARRNUM THEN CALL NUMDISP(ARRPOS):: GOTO
    800
930 DISPLAY AT(23,1)BEEP:"THERE ARE NO MORE
    QUESTIONS!":: DISPLAY AT(24,1)SIZE(28)
940 CALL DELAY(600)
950 CALL EDITCMDS
960 CALL BEEP :: ARRPOS=ARRPOS-1 :: GOTO 840
970 REM
980 REM  GET PREVIOUS RECORD
990 REM
1000 IF ARRPOS>=1 THEN CALL NUMDISP(ARRPOS):: GOTO 800
1010  DISPLAY AT(23,1)BEEP:"CAN'T GO BELOW RECORD #1!"
    :: DISPLAY AT(24,1)SIZE(28)
1020 CALL DELAY(600):: CALL EDITCMDS
1030 CALL BEEP :: ARRPOS=ARRPOS+1 :: GOTO 840
1040 REM
1050 REM  DELETE THE RECORD
1060 REM
1070 DISPLAY AT(23,1)BEEP:"DELETE THIS RECORD? (Y/N)"
    :: DISPLAY AT(24,1)SIZE(28)
1080 CALL KEY(0,K,S):: IF S<=0 THEN 1080
1090 CALL HCHAR(23,30,K):: IF K<>78 AND K<>89 AND
    K<>110 AND K<>121 THEN CALL BEEP :: GOTO 1080
1100 IF K=78 OR K=110 THEN 830
1110 IF ARRNUM-1=0 THEN ARRNUM=0 :: GOTO 750
1120 IF ARRPOS<>ARRNUM THEN INPUT #1,REC ARRNUM:QA$ ::
    PRINT #1,REC ARRPOS:QA$ ELSE ARRPOS=ARRPOS-1
1130 ARRNUM=ARRNUM-1 :: IF ARRNUM<1 THEN 750 ELSE 790
1140 REM
1150 REM   LIST RECORDS TO PRINTER
1160 REM
1170 DISPLAY AT(22,1)SIZE(28):" ENTER BEG. REC #" ::
    DISPLAY AT(23,1)SIZE(28)::DISPLAY AT(24,1)SIZE(28)
1180 ACCEPT AT(22,18)VALIDATE(DIGIT)SIZE(3)BEEP:BRN ::
    IF BRN<1 OR BRN>ARRNUM THEN 1180
1190 DISPLAY AT(23,1)SIZE(28)BEEP:" ENTER END. REC #"
1200 ACCEPT AT(23,18)VALIDATE(DIGIT)SIZE(3)BEEP:ERN ::
    IF ERN<1 OR ERN<BRN OR ERN>ARRNUM THEN 1200
1210 DISPLAY AT(22,1)SIZE(28)BEEP:" TURN ON YOUR
    PRINTER" :: CALL ENTER
1220 DISPLAY AT(22,1)SIZE(28):: DISPLAY  AT
```

```
    (23,1)SIZE(28):" PRINTING RECORDS..."
1230 ON ERROR 1330
1240 OPEN #2:"RS232.BA=9600.DA=8"
1250 FOR I=BRN TO ERN :: INPUT #1,REC I:TEM$
1260 PRINT #2:"RECORD #";I :: PRINT #2 :: PRINT #2:
    "QUESTION: ";SEG$(TEM$,1,26)
1270 PRINT #2:"               ";SEG$(TEM$,27,26)::
    PRINT#2:"              ";SEG$(TEM$,53,26)::
    PRINT #2
1280 PRINT #2 :: PRINT #2:"    ANSWER:   ";
    SEG$(TEM$,79,26)
1290 PRINT #2 :: PRINT #2 :: PRINT #2 :: NEXT I ::
    CLOSE #2 :: GOTO 830
1300 REM
1310 REM  ERROR ROUTINE FOR PRINTING RECORDS
1320 REM
1330 DISPLAY AT(23,1)SIZE(28)BEEP:"COULD NOT PRINT
    RECORDS!" :: CALL DELAY(400):: RETURN 830
1340 REM
1350 REM  SAVING RECORD Ø
1360 REM
1370 ON ERROR 1860
1380 OPEN #1:"DSK1.TRIVIA",INTERNAL,RELATIVE,FIXED 128
1390 PRINT #1,REC Ø:NUMQ
1400 CLOSE #1 :: RETURN
1410 REM
1420 REM  CHANGE FILES/EXIT PROGRAM
1430 REM
1440 NUMQ=ARRNUM :: CLOSE #1 :: GOSUB 1370 :: IF SL=4
    THEN CALL CLEAR :: END
1450 REM
1460 REM  START OF DB
1470 REM
1480 NUMQ=Ø :: CALL CLEAR :: CALL SCREEN(4):: DISPLAY
    AT(1,8)BEEP:"** TRIVIA DB **"
1490 DISPLAY AT(5,2):"DO YOU WANT TO:"
1500 DISPLAY AT(8,3):"1. LOAD AN EXISTING FILE"
1510 DISPLAY AT(10,3):"2. CREATE A NEW FILE"
1520 DISPLAY AT(13,3):"PLEASE SELECT (1-2)"
1530 CALL KEY(Ø,K,S):: IF S<=Ø THEN 1530
1540 DISPLAY AT(13,23):CHR$(K):: IF (K<49)OR(K>5Ø)THEN
    DISPLAY AT(13,23)BEEP SIZE(1):: GOTO 1530
1550 IF K=5Ø THEN 1670
1560 OPNF=FALSE
1570 CALL CLEAR :: DISPLAY AT(5,2)BEEP:"PLEASE INSERT
    THE TRIVIA"
1580 DISPLAY AT(7,2):"DB DISKETTE, INTO YOUR" ::
    DISPLAY AT(9,2):"DISK DRIVE."
1590 CALL ENTER :: CALL CLEAR
1600 ON ERROR 1570
1610 OPEN #1:"DSK1.TRIVIA",INTERNAL,RELATIVE,FIXED 128
1620 INPUT #1,REC Ø:NUMQ
1630 CLOSE #1 :: GOTO 1710
1640 REM
1650 REM  DB-MAIN MENU
1660 REM
```

```
1670 DISPLAY AT(7,2)ERASE ALL BEEP:"PLEASE INSERT AN"
   :: DISPLAY AT(10,2):"INITIALIZED DISKETTE INTO"
1680 DISPLAY AT(13,2):"YOUR DISK DRIVE."
1690 OPNF=TRUE
1700 CALL ENTER :: NUMQ=0 :: GOSUB 1370
1710 ARRNUM=NUMQ
1720 OPEN #1:"DSK1.TRIVIA",INTERNAL,RELATIVE,FIXED 128
1730 DISPLAY AT(4,3)ERASE ALL BEEP:"** TRIVIA DB MAIN
   MENU **"
1740 DISPLAY AT(8,6):"1. ADD QUESTIONS"
1750 DISPLAY AT(10,6):"2. DISPLAY/EDIT" :: DISPLAY  AT
   (11,9):"QUESTIONS"
1760 DISPLAY AT(13,6):"3. CHANGE FILES"
1770 DISPLAY AT(15,6):"4. EXIT PROGRAM" :: DISPLAY  AT
   (18,6):"PLEASE SELECT (1-4)"
1780 CALL KEY(0,K,S):: IF S<=0 THEN 1780
1790 DISPLAY AT(18,26):CHR$(K)
1800 IF (K>48)AND(K<53)THEN K=K-48 :: SL=K ELSE
   DISPLAY AT(18,26)BEEP SIZE(1)::GOTO 1780
1810 ON K GOSUB 210,740,1440,1440
1820 GOTO 1730
1830 REM
1840 REM   ERROR ROUTINE
1850 REM
1860 IF OPNF THEN 1910
1870 CALL ERR(EC,ET,ES,EL)
1880 DISPLAY AT(1,5)ERASE ALL BEEP:"**  FILE ERROR
   **":: DISPLAY AT(6,2):" PLEASE INSERT THE TRIVIA"
1890 DISPLAY AT(8,2):"DB DISKETTE, INTO YOUR" ::
   DISPLAY AT(10,2):"DISK DRIVE.":: CALL ENTER
1900 RETURN 1370
1910 DISPLAY AT(4,2)ERASE ALL BEEP:" PLEASE INITIALIZE
   A " :: DISPLAY AT(6,2):"DISKETTE, THEN RE-RUN THE"
1920 DISPLAY AT(8,2):"PROGRAM." :: CALL ENTER :: CALL
   CLEAR :: END
1930 REM
1940 REM  SUBROUTINES
1950 REM
1960 SUB SCRFORM(NUMREC,TT$)
1970 CALL CLEAR :: CALL OUTLINE
1980 DISPLAY AT(1,3)BEEP:TT$
1990 DISPLAY AT(5,2):"QUESTION#";NUMREC
2000 CALL HCHAR(8,4,128,26):: CALL  HCHAR
   (10,4,128,26)::CALL HCHAR(12,4,128,26)
2010 DISPLAY AT(15,2):"ANSWER FOR QUESTION#";NUMREC ::
   CALL HCHAR(18,4,128,26)
2020 DISPLAY AT(19,6):"(USE ONLY 2 WORDS)"
2030 SUBEND
2040 SUB GETCHAR(STROW,STCOL,NROWS,ANS$,FLG,SV,EX)
2050 ROW=STROW :: COL=STCOL :: NR=1 :: ANS$=""
2060 CALL HCHAR(ROW,COL+2,129):: CALL KEY(5,K,S)
2070 IF S=0 THEN 2060
2080 IF K=146 THEN FLG=1 :: GOTO 2320
2090 IF (K=147)AND SV THEN FLG=2 :: GOTO 2300 ELSE IF
   K=147 THEN 2060
2100 IF (K=152)AND EX THEN FLG=3 :: GOTO 2320 ELSE IF
```

```
    K=152 THEN 2060
2110 IF (K<>13)AND(K<>8)AND((K<32)OR(K>126))THEN CALL
    BEEP :: GOTO 2060
2120 IF K=13 THEN 2300 ELSE IF K=8 THEN 2180
2130 ANS$=ANS$&CHR$(K)
2140 DISPLAY AT(ROW,COL):CHR$(K):: COL=COL+1
2150 IF COL<=27 THEN 2060
2160 ROW=ROW+2 :: NR=NR+1 :: IF NR>NROWS THEN 2210
2170 COL=STCOL :: GOTO 2060
2180 COL=COL-1 :: IF (COL<=STCOL)AND(NR=1)THEN CALL
    HCHAR(ROW,COL+3,32):: CALL BEEP :: GOTO 2050
2190 IF COL<STCOL THEN CALL HCHAR(ROW,COL+3,32)::
    COL=27 :: NR=NR-1 :: ROW=ROW-2
2200 CALL HCHAR(ROW,COL+3,32)::  ANS$=
    SEG$(ANS$,1,LEN(ANS$)-1):: GOTO 2060
2210 ROW=ROW-2 :: NR=NR-1 :: CALL HCHAR(ROW,30,129)
2220 CALL KEY(0,K,S)
2230 IF S<=0 THEN 2220
2240 IF K=146 THEN FLG=1 :: GOTO 2320
2250 IF (K=147)AND SV THEN FLG=2 :: GOTO 2300 ELSE IF
    K=147 THEN 2220
2260 IF (K=152)AND EX THEN FLG=3 :: GOTO 2320 ELSE IF
    K=152 THEN 2220
2270 IF (K<>13)AND(K<>8)THEN CALL BEEP :: GOTO 2220
2280 IF K=13 THEN 2300
2290 GOTO 2180
2300 IF ANS$="" THEN CALL BEEP :: GOTO 2060
2310 CALL HCHAR(ROW,COL+2,32)
2320 SUBEND
2330 SUB BEEP
2340 CALL SOUND(100,523,0)
2350 SUBEND
2360 SUB OUTLINE
2370 CALL HCHAR(3,3,136,29):: CALL  HCHAR
    (21,3,136,29)::CALL VCHAR(3,3,136,18)::CALL VCHAR
    (3,31,136,18)
2380 SUBEND
2390 SUB DELAY(TIME)
2400 FOR I=1 TO TIME :: NEXT I
2410 SUBEND
2420 SUB ENTER
2430 DISPLAY AT(23,2):"PRESS <ENTER> TO CONTINUE"
2440 CALL KEY(0,K,S):: IF S<=0 THEN 2440
2450 IF K<>13 THEN 2440
2460 SUBEND
2470 SUB PARSEANS(ANS$,ERR)
2480 WD$="" :: WRDS=0
2490 FOR I=1 TO LEN(ANS$)
2500 IF ASC(SEG$(ANS$,I,1))=32 AND WD$="" THEN 2540
2510 IF SEG$(ANS$,I,1)=" " AND SEG$(ANS$,I+1,1)=" "
    THEN 2540
2520 IF ASC(SEG$(ANS$,I,1))=32 OR I=LEN(ANS$)THEN
    WRDS=WRDS+1 :: IF WRDS>2 THEN ERR=-1 :: GOTO 2620
2530 WD$=WD$&SEG$(ANS$,I,1)
2540 NEXT I
2550 T$="" :: FOR J=1 TO LEN(WD$):: A=ASC
```

```
    (SEG$(WD$,J,1))
2560 IF A>96 AND A>123 THEN A=A-32
2570 T$=T$&CHR$(A):: NEXT J
2580 FOR I=1 TO LEN(T$):: IF ASC(SEG$(T$,I,1))=32 AND
    I<>1 THEN 2600
2590 NEXT I :: GOTO 2620
2600 A$=SEG$(T$,1,I):: IF (A$="A " OR A$="THE " OR
    A$="AN ")AND I<>LEN(T$)THEN T$=SEG$(T$,I+1,LEN(T$)
    I)
2610 ANS$=T$
2620 SUBEND
2630 SUB ERASEQUEST
2640 DISPLAY AT(7,2)SIZE(26):: DISPLAY AT(9,2)SIZE(26)
2650 DISPLAY AT(11,2)SIZE(26)
2660 SUBEND
2670 SUB ERASEANS
2680 DISPLAY AT(17,2)SIZE(26)
2690 SUBEND
2700 SUB FULL
2710 CALL CLEAR :: CALL BEEP :: CALL BEEP
2720 DISPLAY AT(12,2):"DISKETTE IS FULL!!" :: CALL
    ENTER
2730 SUBEND
2740 SUB ADDCMDS
2750 DISPLAY AT(22,1)SIZE(28)
2760 DISPLAY AT(23,1)SIZE(28):"C/S-SAVE, C/R-REDO, C/X
    EXIT"
2770 DISPLAY AT(24,1)SIZE(28)
2780 SUBEND
2790 SUB PRRNT
2800 DISPLAY AT(22,1)SIZE(28)
2810 DISPLAY AT(23,1):" PRESS <ENTER> WHEN DONE!"
2820 DISPLAY AT(24,1)SIZE(28)
2830 CALL BEEP
2840 SUBEND
2850 SUB ERRMSG
2860 DISPLAY AT(20,7)BEEP:"RE-ENTER ANSWER" :: CALL
    DELAY(400)
2870 DISPLAY AT(20,7)SIZE(17):: DISPLAY  AT
    (17,2)SIZE(26)
2880 SUBEND
2890 SUB EDITCMDS
2900 DISPLAY AT(22,2):"C/R-REDO, C/N-NEXT, C/P-PREVC/D
    DEL, C/L-LIST, C/X-EXIT"
2910 SUBEND
2920 SUB NUMDISP(ARRCTR)
2930 DISPLAY AT(5,11)SIZE(4):ARRCTR :: DISPLAY  AT
    (15,22)SIZE(4):ARRCTR
2940 SUBEND
2950 SUB WRAP(Q$,BLK$)
2960 WL=0
2970 WL=WL+1 :: IF WL>2 THEN 3030
2980 IF ASC(SEG$(Q$,WL*26,1))=32 THEN 2970
2990 ST=WL*26-1 :: EN=(WL-1)*26+1 :: FOR I=ST TO EN
    STEP -1 :: IF ASC(SEG$(Q$,I,1))=32 THEN 3010
3000 NEXT I :: GOTO 2970
```

```
3010 T$=SEG$(Q$,I+1,WL*26-I):: T2$=SEG$(SEG$(SEG$
    (Q$,1,I)&BLK$,1,WL*26)&T$&SEG$(Q$,WL*26+1,
    (WL+1)*26),1,78)
3020 Q$=T2$ :: GOTO 2970
3030 DISPLAY AT(7,2):SEG$(Q$,1,26):: DISPLAY AT
    (9,2):SEG$(Q$,27,26)
3040 DISPLAY AT(11,2):SEG$(Q$,53,26)
3050 SUBEND
```

Variable Listing — Main Program

A$	Temporarily holds the answer before packing and concatenating with the question.
ANS$	Holds the answer. Maximum length is 26 characters.
ARRNUM	Holds the total number of questions that are on the disk as well as the next record number to place the question and answer.
ARRPOS	Holds the current record pointer for the disk. Points to the record to load into memory. Used in Display/Edit Menu.
BLK$	Blank string the size of the maximum size field (78). Used for packing the question and answer to their maximum length.
BOTH	Boolean flag set to indicate if the user wants to REDO both the question and the answer. If true, the user wants to edit both parts of the record.
BRN	Beginning record number to print to the printer.
CHNGS	Boolean flag. Indicates if any changes have been made to the existing file in memory.
DE	Flag set to indicate whether the program execution came from the Display/Edit menu. If true, then it did. Make sure you return to Display/Edit menu.
ERN	Ending record number to print to the printer.

ERR	Flag used to indicate if an error occurred within a procedure. This type of an error is not a system error but a user error (invalid response or illegal characters, etc.).
EX	Boolean flag. Indicates whether or not to allow the user to use the C/X key.
FALSE	Boolean flag set to 0. Indicates a false value.
FLAG	Flag set to indicate if one of the command keys (C/S, C/X, etc.) has been pressed. Returns a 1 if C/R pressed, a 2 if C/S is pressed, and a 3 if C/X has been pressed.
I	Looping variable for FOR...NEXT statements.
K	Used in the CALL KEY routine to hold the ASCII value of the key that is pressed.
NUMQ	Number of questions stored on disk.
OPNF	Open flag indicates if the file is in open mode or if it is closed.
Q$	Holds the question. Maximum length is 78 characters.
QA$	Holds the questions and answers in a concatenated string. Total string length is 104 characters, 78 for the question and 26 for the answer.
S	Used in the CALL KEY routine to hold the status of the keyboard.
SL	Holds the selection number the user chose from the Main Menu.
SV	Boolean flag. Indicates whether or not to allow the user to press C/S while typing in the question or answer.
TEM$	Holds the temporary string that is brought into memory for listing to the printer.

TOT	Temporarily holds the total number of questions and answers that are on the disk.
TRUE	Boolean flag indicates that the result is indeed true. Usually a −1, but the TI-99/4A accepts any number other than 0 to be true.
TT$	Holds the title string to be printed at the top of each screen.

Variable Listing for SUBROUTINES

SUB SCRFORM(NUMREC,TT$)

NUMREC	Record number to print. Same as the question number. Shows what question the user is currently entering.
TT$	Same purpose, as TT$ is in the MAIN PROGRAM.

SUB GETCHAR(STROW,STCOL,NROWS,ANS$,FLG,SV,EX)

STROW	Starting row location to place the cursor and where to start printing characters as they are typed.
STCOL	Starting column location for placing the cursor and displaying the characters.
NROWS	The number of rows the cursor is allowed to travel. For the question, NROWS is set to 3; for the answer, it is set to 1.
ANS$	Holds the answer or the question as the user types in the characters.
FLG	Flag returns a 1,2, or 3 depending on which, if any, of the three command keys (C/S, C/R, C/X) was pressed.
SV	Tells subroutine whether or not to allow the user to press C/S as a valid key.
EX	Tells subroutine whether or not to accept C/X as a valid input.

ROW	Is used for keeping the cursor on the same line until it reaches the end of the line, then move it down to the next line.
COL	Used for keeping the cursor as well as the characters placed one in front of the other.
NR	Counter to keep track of how many rows the cursor has traveled compared to how many it is allowed to move.
K & S	Same purpose as in the MAIN PROGRAM.

SUB DELAY(TIME)

TIME	Factor for how long the FOR...NEXT loop should go for a delaying time period.
I	Same purpose as in the MAIN PROGRAM.

SUB ENTER

K & S	Same purpose as in the MAIN PROGRAM.

SUB PARSEANS(ANS$,ERR)

ANS$	Holds the answer that you want parsed. Parses out the words THE, A, and AN from the answer. Returns the correct answer in this variable.
ERR	Is set if the answer contains more than two words.
WD$	Temporarily holds the new answer after parsing.
WRDS	Counter used to count how many words are in your answer.
I & J	Looping variables.
T$	Temporarily holds the answer after it is converted to all uppercase letters.
A	Holds the ASCII value of each character in the answer while converting the character to uppercase.
A$	Holds the first word of the answer to see if it matches A, AN, or THE.

SUB NUMDISP(ARRCTR)

ARRCTR

The question number that is displayed on the screen. Indicates the current question.

SUB WRAP(Q$,BLK$)

Q$	Holds the question that is to be wrapped around the screen.
BLK$	Same purpose as in the MAIN PROGRAM.
WL	Indicates Which Line we are currently calculating for size.
ST	Starting character position within the string Q$.
EN	Ending position within the Question string.
I	Same as in MAIN PROGRAM.
T$	Temporarily holds the new reformatted question (1st part).
T2$	Temporarily holds the rest of the string in its newly reformatted state.

Appendix D
The Diskette Game

```
10 REM
20 REM TRIVIA GAME - DISKETTE VERSION
30 REM
40 CALL INIT :: CALL LOAD(-31806,16)
50 CALL CLEAR :: CALL SCREEN(4):: OPTION BASE 1 :: DIM
   QA$(75),RQN(75),N(75),ND(9),NT(5)
60 BLK$=RPT$(" ",26):: FALSE=0 :: TRUE=NOT FALSE
70 CALL CHAR(128,"FF0000000000000"):: CALL
   CHAR(129,"00007E7E7E7E0000")
80 CALL CHAR(136,"FFFFFFFFFFFFFFFF"):: CALL
   COLOR(14,13,4)
90 GOTO 470
100 REM
110 REM DISPLAYS STATUS OF CURRENT PLAYER
120 REM
130 DISPLAY AT(19,2):"PLAYER#";PLYR :: DISPLAY AT
   (19,15):"NAME: ";SEG$(PN$(PLYR),1,6)
140 DISPLAY AT(21,2):"SCORE:";SC(PLYR):: DISPLAY AT
   (21,15):"NC/NQ:";NC(PLYR);"/";NQ(PLYR)
150 RETURN
160 REM
170 REM  DISPLAY STATUS OF ALL PLAYER AT THE END OF
   THE GAME
180 REM
190 CALL SOUND(300,-1,0):: CALL SOUND(300,-1,0)::
   DISPLAY AT(1,10)ERASE ALL:"FINAL STATS"
200 DISPLAY AT(5,2):"NAME        SCORE   NC/NQ" ::
   CALL HCHAR(6,4,128,29)
210 IF NP=1 THEN DISPLAY AT(8,2):PN$(1):: DISPLAY  AT
   (8,14):SC(1):: DISPLAY AT(8,22):NC(1);"/";NQ(1)::
   GOTO 310
220 FOR I=1 TO NP :: PN(I)=I :: NEXT I :: FOR J=1 TO
   NP-1 :: J1=J
230 FOR K=J+1 TO NP :: IF SC(J1)<SC(K)THEN J1=K
240 NEXT K :: IF J1<>J THEN T=SC(J):: SC(J)=SC(J1)::
   SC(J1)=T :: T=PN(J):: PN(J)=PN(J1):: PN(J1)=T
250 NEXT J
260 FOR I=1 TO NP :: DISPLAY AT(6+I,2):PN$(PN(I))::
   DISPLAY AT(6+I,14):SC(I)
270 DISPLAY AT(6+I,22):NC(PN(I));"/";NQ(PN(I)):: NEXT
   I
280 REM
290 REM  PRINT FINAL STATS?
300 REM
310 DISPLAY AT(22,2)SIZE(28)BEEP:"PRINT FINAL STATS?
   (Y/N)" :: YFLG=FALSE :: CALL GETYN(YFLG):: IF NOT
   YFLG THEN 400
```

```
320 DISPLAY AT(22,1)SIZE(28):" PRINTING STATS..."
330 ON ERROR 1730
340 OPEN #2:"RS232.BA=9600.DA=8"
350 PRINT #2:"     **  FINAL STATS  **" :: PRINT #2 ::
    PRINT #2
360 PRINT #2:"NAME                SCORE        NC/NQ" ::
    PRINT #2:"-----------------------------------" ::
    PRINT #2
370 IF NP=1 THEN PRINT #2:PN$(1);"           ";
    SC(1);"          ";NC(1);"/";NQ(1):: GOTO 390
380 FOR I=1 TO NP :: PRINT #2:PN$(PN(I));"           "
    ;SC(I);"       ";NC(PN(I));"/";NQ(PN(I)):: NEXT I
390 CLOSE #2
400 IF QUIT THEN CALL DELAY(800):: GOTO 430
410 DISPLAY AT(22,2)BEEP:"ANOTHER GAME? (Y/N)" ::
    YFLG=FALSE :: CALL GETYN(YFLG)
420 IF YFLG THEN DISPLAY AT(10,2)ERASE ALL:"ONE MOMENT
    PLEASE..." :: RUN 20
430 DISPLAY AT(10,2)ERASE ALL:"HAVE A NICE DAY!" ::
    END
440 REM
450 REM  STARTS GAME
460 REM
470 DISPLAY AT(6,4)BEEP:"WELCOME TO THE TI-99/4A" ::
    DISPLAY AT(10,6):"**  TRIVIA GAME  **"
480 DISPLAY AT(22,2):"NEED INSTRUCTIONS? (Y/N)" ::
    YFLG=FALSE :: CALL GETYN(YFLG)
490 IF NOT YFLG THEN 680
500 DISPLAY AT(1,2)ERASE ALL BEEP:"  ** INSTRUCTIONS
    **    PG 1"
510 DISPLAY AT(3,2):"  THE OBJECT OF THE GAME IS" ::
    DISPLAY AT(4,2):"TO GAIN AS MANY POINTS AS"
520 DISPLAY AT(5,2):"POSSIBLE BY ANSWERING" :: DISPLAY
    AT(6,2):"RANDOM TRIVIA QUESTIONS."
530 DISPLAY AT(7,2):"EACH QUESTION IS WORTH A" ::
    DISPLAY AT(8,2):"STARTING VALUE OF 25"
540 DISPLAY AT(9,2):"POINTS." :: DISPLAY AT(11,2):"
    THE FASTER THE PLAYER" :: DISPLAY AT (12,2)
    :"ANSWERS THE QUESTION, THE"
550 DISPLAY AT(13,2):"MORE POINTS THEY CAN" :: DISPLAY
    AT(14,2):"OBTAIN.  AS TIME GOES BY,"
560 DISPLAY AT(15,2):"THE POINT VALUE DECREASES." ::
    DISPLAY AT(17,2):"*  ALL ANSWERS MUST MATCH"
570 DISPLAY AT(18,2):"EXACTLY, IN ORDER TO" :: DISPLAY
     AT(19,2):"OBTAIN ANY POINTS!" :: CALL ENTER
580 DISPLAY AT(1,2)ERASE ALL BEEP:"  ** INSTRUCTIONS
    **    PG 2" :: DISPLAY AT(3,2):"  IF THE PLAYER
    MISSES THE"
590 DISPLAY AT(4,2):"ANSWER, THE NEXT PLAYER IN" ::
    DISPLAY AT(5,2):"LINE GETS A CHANCE TO"
600 DISPLAY AT(6,2):"ANSWER THE QUESTION WITH" ::
    DISPLAY AT(7,2):"THE POINT VALUE STARTING AT"
610 DISPLAY AT(8,2):"12.  ONCE ALL THE PLAYERS" ::
    DISPLAY AT(9,2):"HAVE HAD A CHANCE TO ANSWER"
620 DISPLAY AT(10,2):"THE QUESTION, THE CORRECT" ::
    DISPLAY AT(11,2):"ANSWER IS THEN DISPLAYED."
```

```
630 DISPLAY AT(13,2):"  THE GAME CAN BE PLAYED BY" ::
    DISPLAY AT(14,2):"1-4 PLAYERS."
640 DISPLAY AT(18,2):"   GOOD LUCK!...HAVE FUN!" ::
    CALL ENTER
650 REM
660 REM   TELLS USER TO INSERT TRIVIA DISKETTE
670 REM
680 DISPLAY AT(4,2)ERASE ALL BEEP:"PLEASE INSERT THE
    TRIVIA" :: DISPLAY AT(6,2):"DB DISKETTE INTO YOUR"
690 DISPLAY AT(8,2):"DISK DRIVE." :: CALL ENTER
700 ON ERROR 1650
710 OPEN #1:"DSK1.TRIVIA",INTERNAL,RELATIVE,FIXED 128
720 INPUT #1,REC 0:NUMQ
730 IF NUMQ>75 THEN 750
740 FOR I=1 TO NUMQ :: RQN(I)=I :: NEXT I :: GOTO 940
750 DISPLAY AT(8,2)ERASE ALL BEEP:"PLEASE WAIT...I'M
    WORKING"
760 RANDOMIZE :: FOR I=1 TO 75
770 WN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
    CKNUM(WN,RQN(),ERR,75):: IF ERR THEN 770 ELSE
    RQN(I)=WN
780 NEXT I
790 REM
800 REM  SORT NUMBERS
810 REM
820 P=1 :: Q=75 :: T0=0
830 IF P>=Q THEN 920
840 V=RQN(P):: I=P :: J=Q+1
850 J=J-1 :: IF RQN(J)>V THEN 850
860 I=I+1 :: IF RQN(I)<V AND I<75 THEN 860
870 IF J>I THEN T=RQN(I):: RQN(I)=RQN(J):: RQN(J)=T ::
    GOTO 850
880 RQN(P)=RQN(J):: RQN(J)=V
890 IF (J-P)<(Q-J)THEN N(T0+1)=J+1 :: N(T0+2)=Q ::
    Q=J-1 :: GOTO 910
900 N(T0+1)=P :: N(T0+2)=J-1 :: P=J+1
910 T0=T0+2 :: GOTO 830
920 IF T0<>0 THEN Q=N(T0):: P=N(T0-1):: T0=T0-2 ::
    GOTO 830
930 FOR I=1 TO 75 :: N(I)=0 :: NEXT I
940 DISPLAY AT(2,2)ERASE ALL BEEP:" PLEASE LEAVE THE
    DISKETTE" :: DISPLAY AT(4,2):"IN YOUR DISK DRIVE."
950 DISPLAY AT(6,2):"I WILL NOW LOAD THE" :: DISPLAY
    AT(8,2):"QUESTIONS FROM YOUR TRIVIA"
960 DISPLAY AT(10,2):"DISKETTE." :: CALL ENTER
970 DISPLAY AT(10,4)ERASE ALL BEEP:"LOADING
    QUESTIONS..."
980 REM
990 REM  LOADS RECORDS
1000 REM
1010 ON ERROR 1650
1020 IF NUMQ>75 THEN NUMQ=75
1030 FOR I=1 TO NUMQ :: INPUT #1,REC RQN(I):QA$(I)::
    NEXT I :: CLOSE #1
1040 DISPLAY AT(6,2)ERASE ALL BEEP:"PLEASE WAIT..." ::
    DISPLAY AT(10,2):"SCRAMBLING QUESTIONS"
```

```
105Ø RANDOMIZE
1060 NUMDBL=INT(NUMQ*.11)+1 :: NUMTPL=INT(NUMQ*.04)+1
1070 FOR I=1 TO NUMQ
1080 WN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
     CKNUM(WN,N(),ERR,NUMQ)
1090 IF ERR THEN 1080 ELSE N(I)=WN
1100 NEXT I :: FOR I=1 TO NUMDBL
1110 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
     CKNUM(QN,ND(),ERR,9)
1120 IF ERR THEN 1110 ELSE ND(I)=QN
1130 NEXT I :: FOR I=1 TO NUMTPL
1140 QN=INT(RND*NUMQ)+1 :: ERR=FALSE :: CALL
     CKNUM(QN,NT(),ERR,4):: IF ERR THEN 1140
1150 CALL CKNUM(QN,ND(),ERR,9):: IF ERR THEN 1140
1160 NT(I)=QN :: NEXT I
1170 DISPLAY AT(10,2)ERASE ALL BEEP:"NUMBER OF PLAYERS
     (1-4)"
1180 CALL KEY(0,K,S):: IF S<=0 THEN 1180
1190 CALL HCHAR(10,28,K):: IF K<49 OR K>52 THEN CALL
     BEEP :: CALL HCHAR(10,28,32):: GOTO 1180
1200 NP=K-48 :: FOR I=1 TO NP :: DISPLAY AT(10,2)ERASE
     ALL:"PLAYER#";I;"NAME:"
1210 ACCEPT AT(10,18)VALIDATE(UALPHA)BEEP
     SIZE(11):PN$(I):: IF PN$(I)="" THEN 1210
1220 PN$(I)=SEG$(PN$(I)&BLK$,1,10)
1230 NEXT I
1240 REM
1250 REM  STARTS GAME
1260 REM
1270 DISPLAY AT(1,5)ERASE ALL BEEP:"**  TRIVIA GAME
     **" :: CALL HCHAR(2,3,136,29)
1280 CALL HCHAR(18,3,136,29):: CALL
     VCHAR(2,3,136,16)::CALL VCHAR(2,31,136,16):: CALL
     VCHAR(18,3,136,6)
1290 CALL VCHAR(18,31,136,6):: CALL HCHAR(24,3,136,29)
1300  DISPLAY AT(17,6)SIZE(20):"(USE ONLY 2 WORDS)"::
     DISPLAY AT(5,2):"QUESTION#"
1310 DISPLAY AT(14,2):"ANSWER:" :: CALL
     HCHAR(16,4,128,26)
1320 DISPLAY AT(23,6):"CTRL-Q TO QUIT" :: DBL=FALSE ::
     TPL=FALSE
1330 PLYR=1 :: ARRLOC=1 :: FOR I=1 TO NP :: NQ(I)=0 ::
     NC(I)=0 :: SC(I)=0 :: NEXT I
1340 TIM=0 :: CALL TIMER(TIM):: GOSUB 130 :: DISPLAY
     AT(5,13)SIZE(2):ARRLOC
1350 DISPLAY AT(7,2):SEG$(QA$(N(ARRLOC)),1,26)::
     DISPLAY AT(9,2):SEG$(QA$(N(ARRLOC)),27,26)
1360 DISPLAY AT(11,2):SEG$(QA$(N(ARRLOC)),53,26)::
     STPLYR=PLYR
1370 FOR I=1 TO NUMDBL :: IF ND(I)=N(ARRLOC)THEN
     DBL=TRUE :: GOTO 1400
1380 NEXT I :: DBL=FALSE :: FOR I=1 TO NUMTPL :: IF
     NT(I)=N(ARRLOC)THEN TPL=TRUE:: GOTO 1400
1390 NEXT I :: TPL=FALSE
1400 DISPLAY AT(15,2)SIZE(26):: DISPLAY  AT
     (12,9)SIZE(15)
```

```
1410 IF DBL THEN CALL BEEP :: DISPLAY  AT
    (12,10):"DOUBLE VALUE" :: CALL BEEP
1420 IF TPL THEN CALL BEEP :: DISPLAY AT
    (12,10):"TRIPLE VALUE" :: CALL BEEP
1430 TU=FALSE :: ANS$="" :: QUIT=FALSE :: CALL
    GETANS(ANS$,15,2,26,QUIT,TIM,TU)
1440 POINTS=25-TIM :: IF QUIT THEN 1600
1450 IF TU THEN 1490
1460 IF DBL THEN POINTS=POINTS*2
1470 IF TPL THEN POINTS=POINTS*3
1480 CALL PARSEANS(ANS$):: ANS$=SEG$(ANS$&BLK$,1,26)::
    CA$=SEG$(QA$(N(ARRLOC)),79,26):: CALL CONVERTUPPER
    (CA$):: IF CA$=ANS$ THEN 1540
1490 CALL SOUND(700,-3,0):: CALL DELAY(200):: NQ(PLYR)
    =NQ(PLYR)+1
1500 PLYR=PLYR+1 :: IF PLYR>NP THEN PLYR=1
1510 IF PLYR<>STPLYR THEN TIM=13 :: CALL TIMER(TIM)::
    GOSUB 130 :: GOTO 1400
1520 DISPLAY AT(14,2)BEEP:"THE CORRECT ANSWER IS:" ::
    DISPLAY AT(15,2):SEG$(QA$(N(ARRLOC)),79,26)
1530 CALL DELAY(500):: DISPLAY AT
    (14,2)SIZE(26):"ANSWER:" :: DISPLAY AT(15,2)
    SIZE(26):: GOTO 1550
1540 CALL SOUND(700,-1,0):: SC(PLYR)=SC(PLYR)+POINTS
    ::NC(PLYR)=NC(PLYR)+1 :: NQ(PLYR)=NQ(PLYR)+1
    ::  GOSUB 130 :: CALL DELAY(300)
1550 ARRLOC=ARRLOC+1 :: IF ARRLOC>NUMQ THEN CALL
    DELAY(200):: GOTO 190 ELSE PLYR=STPLYR+1 :: IF
    PLYR>NP THEN PLYR=1
1560 DBL=FALSE :: TPL=FALSE :: GOTO 1340
1570 REM
1580 REM  QUITS GAME EARLY
1590 REM
1600 DISPLAY AT(23,6)SIZE(20)BEEP:"ARE YOU SURE?
    (Y/N)":: YFLG=FALSE :: CALL GETYN(YFLG)
1610 IF YFLG THEN 190 ELSE DISPLAY AT
    (23,6)SIZE(20):"CTRL-Q TO QUIT" :: GOTO 1430
1620 REM
1630 REM  ERROR ROUTINE
1640 REM
1650 DISPLAY AT(1,5)ERASE ALL BEEP:"**  FILE ERROR
    **":: DISPLAY AT(5,2):"  I HAVE ENCOUNTERED AN"
1660 DISPLAY AT(7,2):"ERROR IN LOADING FROM THE" ::
    DISPLAY AT(9,2):"DISKETTE."
1670 DISPLAY AT(14,2):" PLEASE RE-RUN THE PROGRAM." ::
    DISPLAY AT(22,2):" PRESS <ENTER> TO EXIT"
1680 CALL KEY(0,K,S):: IF S<=0 THEN 1680
1690 IF K<>13 THEN 1680 ELSE CALL CLEAR :: END
1700 REM
1710 REM  ERROR ROUTINE FOR PRINTING FINAL STATS
1720 REM
1730 DISPLAY AT(22,1)SIZE(28)BEEP:"COULDN'T PRINT
    STATS!" :: CALL DELAY(300)
1740 RETURN 310
1750 SUB CONVERTUPPER(WD$)
1760 T$="" :: FOR J=1 TO LEN(WD$)::  A=ASC (SEG$(WD$,
```

```
     J,1)):: IF A>96 AND A<123 THEN A=A-32
1770 T$=T$&CHR$(A):: NEXT J :: WD$=T$
1780 SUBEND
1790 SUB GETANS(ANS$,STROW,STCOL,LN,QUIT,TIMR,TU)
1800 TT=0
1810 ROW=STROW :: COL=STCOL :: ANS$=""
1820 CALL HCHAR(ROW,COL+2,129):: CALL KEY(5,K,S)
1830 IF S=0 AND TT>=15 THEN TT=0 :: TIMR=TIMR+1 ::
     CALL TIMER(TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO
     2060
1840 IF S=0 THEN TT=TT+1 :: GOTO 1820
1850 IF K=145 THEN QUIT=-1 :: GOTO 2030
1860 IF (K<>13)AND(K<>8)AND((K<32)OR(K>126))THEN CALL
     BEEP :: GOTO 1820
1870 TT=TT+1 :: IF TT>=15 THEN TT=0 :: TIMR=TIMR+1 ::
     CALL TIMER(TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO
     2060
1880 IF K=13 THEN 2010 ELSE IF K=8 THEN 1920
1890 ANS$=ANS$&CHR$(K)
1900 DISPLAY AT(ROW,COL):CHR$(K):: COL=COL+1
1910 IF COL<STCOL+LN THEN 1820 ELSE CALL BEEP :: GOTO
     1940
1920 COL=COL-1 :: IF (COL<=STCOL)THEN CALL HCHAR
     (ROW,COL+3,32):: CALL BEEP :: GOTO 1810
1930 ANS$=SEG$(ANS$,1,LEN(ANS$)-1):: CALL HCHAR
     (ROW,COL+3,32):: GOTO 1820
1940 CALL HCHAR(ROW,COL+2,129):: CALL KEY(0,K,S)
1950 IF S<=0 THEN TT=TT+1
1960 IF TT>=15 THEN TT=0 :: TIMR=TIMR+1 :: CALL TIMER
     (TIMR):: IF TIMR>=25 THEN TU=-1 :: GOTO 2060
1970 IF S<=0 THEN TT=TT+1 :: GOTO 1940
1980 IF K=145 THEN QUIT=-1 :: GOTO 2030
1990 IF (K<>13)AND(K<>8)THEN CALL BEEP :: GOTO 1940
2000 IF K=13 THEN 2010 ELSE 1920
2010 IF ANS$="" THEN CALL BEEP :: GOTO 1820
2020 CALL HCHAR(ROW,COL+2,32)
2030 SUBEND
2040 SUB BEEP
2050 CALL SOUND(100,523,0)
2060 SUBEND
2070 SUB PARSEANS(ANS$)
2080 WD$="" :: FOR I=1 TO LEN(ANS$)
2090 IF ASC(SEG$(ANS$,I,1))=32 AND WD$="" THEN 2120
2100 IF SEG$(ANS$,I,1)=" " AND SEG$(ANS$,I+1,1)=" "
     THEN 2120
2110 WD$=WD$&SEG$(ANS$,I,1)
2120 NEXT I :: CALL CONVERTUPPER(WD$)
2130 FOR I=1 TO LEN(WD$):: IF ASC(SEG$(WD$,I,1))=32
     AND I<>1 THEN 2150
2140 NEXT I :: GOTO 2170
2150 A$=SEG$(WD$,1,I):: IF (A$="A " OR A$="THE " OR
     A$="AN ")AND I<>LEN(WD$)THEN WD$=SEG$(WD$,I+1,
     LEN(WD$)-I)
2160 ANS$=WD$
2170 SUBEND
2180 SUB TIMER(SEC)
```

```
2190 DISPLAY AT(3,23)SIZE(4):25-SEC
2200 SUBEND
2210 SUB GETYN(YFLG)
2220 CALL KEY(0,K,S):: IF S<=0 THEN 2220
2230 IF K<>78 AND K<>110 AND K<>89 AND K<>121 THEN
     2220
2240 IF K=78 OR K=110 THEN YFLG=0 ELSE YFLG=-1
2250 SUBEND
2260 SUB ENTER
2270 DISPLAY AT(22,1)SIZE(28):"  PRESS <ENTER> TO
     CONTINUE"
2280 CALL KEY(0,K,S):: IF S<=0 THEN 2280 ELSE IF K<>13
     THEN 2280
2290 SUBEND
2300 SUB CKNUM(NUMB,T(),ERR,MAXLIM)
2310 FOR I=1 TO MAXLIM :: IF NUMB=T(I)THEN ERR=-1
2320 NEXT I
2330 SUBEND
2340 SUB DELAY(DURAT)
2350 FOR K=1 TO DURAT :: NEXT K
2360 SUBEND
```

Variable Listing — Main Program

ANS$	Holds the answer that the player has typed in. Maximum length is 26 characters.
ARRLOC	Holds the array number of the question that is displayed.
BLK$	Blank string the size of the maximum size field (26). Used for packing the answer to its maximum length.
CA$	Holds the correct answer. Used for comparing the two answers for equality.
DBL	Boolean flag to indicate if the current question is to be double value or not. Used for printing the DOUBLE VALUE message.
ERR	Flag used to indicate if an error occurred within a procedure.
FALSE	Boolean flag set to 0. Indicates a false value.
I & J	Looping variable for FOR...NEXT statements.

K	Used in the CALL KEY routine to hold the ASCII value of the key that is pressed. K also used for looping.
N(75)	Holds the random generated sequence of numbers in which to display the questions in a random format.
NC()	Holds the player's number of correct questions that (s)he answered.
ND(9)	Holds the random generated numbers that are to be assigned DOUBLE point values.
NP	The total number of players playing the game, from one to four.
NQ()	Holds the number of questions that each player was asked.
NT(5)	Holds the random generated numbers that are to be assigned TRIPLE point value.
NUMDBL	Holds the number of DOUBLE point questions that this game is supposed to have.
NUMQ	Number of questions currently loaded into memory from the disk file.
NUMTPL	Holds how many questions are to be TRIPLE pointers.
P	Holds the current pointer for the lowest partition using Quicksort routine.
PLYR	Holds the current player number to display on the screen.
PN$()	Holds the players' names up to 10 characters long, but only six are displayed on the game screen. The whole name is displayed at the end of the game in the statistic screen.
Q	Holds the maximum array location to sort.

QA$(75)	Holds the questions and answers in a concatenated string. Total string length is 104 characters, 78 for the question and 26 for the answer.
QN	Holds the random question number for selecting the double and triple value questions without obtaining duplicate numbers.
QUIT	Holds the boolean flag for quitting the game early.
RQN(75)	Holds the **R**andom **Q**uestion **N**umbers to load in memory.
S	Used in the CALL KEY routine to hold the status of the keyboard.
SC()	Holds the player's score.
STPLYR	Holds the player number of who started first. This keeps the players going in the same order no matter who answers the questions from a missed question.
T	Used for holding the number that needs switching.
T0	Holds the stack pointer for Quicksorting.
TIM	Holds the timer value. Starts at 0 and is incremented until either the player answers the questions or it reaches the maximum point limit.
TPL	Holds the TRIPLE value flag set for printing the TRIPLE VALUE message or not.
TRUE	Boolean flag indicates that the result is indeed true. Usually a −1, but the TI-99/4A accepts any number other than 0 to be true.
TU	Boolean flag set only if the timer runs out of time. Indicates that **T**ime is **U**p.

V	Temporarily holds a number to sort on.
WN	Holds a random number to select from the file.
YFLG	Boolean flag to indicate whether the user responded with a YES or a NO.

Variable Listing for SUBROUTINES

SUB CONVERTUPPER(WD$)

WD$	The string that needs to be converted to uppercase characters.
T$	Temporarily holds the new characters as they are converted.
J	Same purpose as in the MAIN PROGRAM.
A	Holds the ASCII value for one character at a time.

SUB GETANS(ANS$,STROW,STCOL,LN,QUIT,TIMR,TU)

ANS$	Holds the answer as the user types.
STROW	Starting row location to place the cursor and where to start printing characters as they are typed.
STCOL	Starting column location for placing the cursor and displaying the characters.
LN	Holds the length of the answer. (26 is the maximum amount.)
QUIT	Is the quit flag. Gets set if the user presses CTRL-Q.
TIMR	Holds the timer value as it clicks away.
TU	Flag that indicates if time ran out on the player before (s)he could answer the question.
TT	Runs the timer from within the subroutine.

ROW	Holds the starting row location for displaying the printed characters.
COL	Holds the starting column number for displaying the characters side by side as they are typed.
K & S	Same purpose as in the MAIN PROGRAM.

SUB DELAY(DURAT)

DURAT	Factor for how long the FOR...NEXT loop should go for a delaying time period.
K	Looping variable.

SUB ENTER

K & S	Same purpose as in the MAIN PROGRAM.

SUB PARSEANS(ANS$)

ANS$	Holds the answer that you want parsed. Parses out the words THE, A, and AN from the answer. Returns the correct answer in this variable.
WD$	Temporarily holds the new answer after parsing.
WRDS	Counter used to count how many words are in your answer.
I & J	Looping variables.
A$	Holds the first word of the answer to see if it matches A, AN, or THE.

SUB GETYN(YFLG)

YFLG	Holds a true value if the user presses a Y, false if the user presses an N. Returns one of the two above to the MAIN PROGRAM for execution on the flag.
K & S	Same purpose as in the MAIN PROGRAM.

SUB TIMER(SEC)

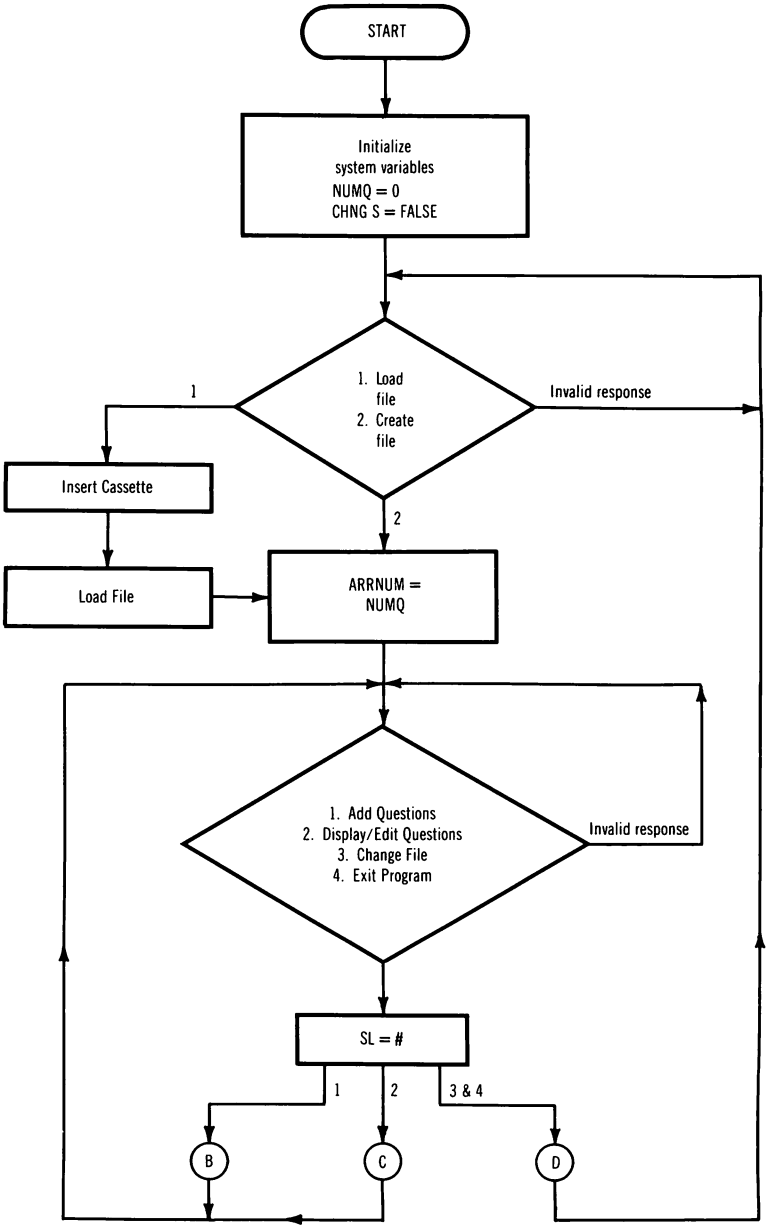
SEC	How many points to decrease the starting value by.

SUB CKNUM(NUMB,T(),ERR,MAXLIM)

NUMB	Number to check for duplication in selection random numbers.
T()	The array to compare the NUMB to, to see if any duplication is found.
ERR	Set to true if the NUMB already exists in the array T().
MAXLIM	Maximum amount of numbers to check in the array.
I	Looping variable.

Flowchart 1 — Cassette Data Base (foldout)

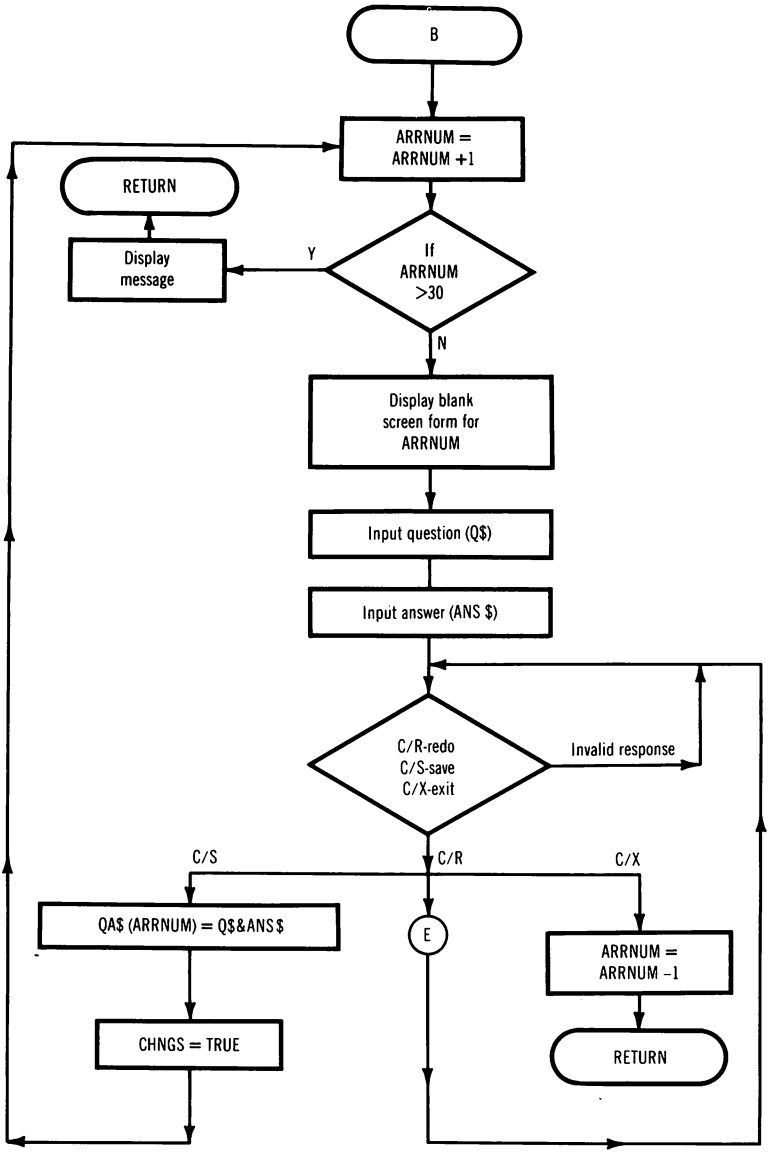
Flowchart 1 — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base

(foldout)

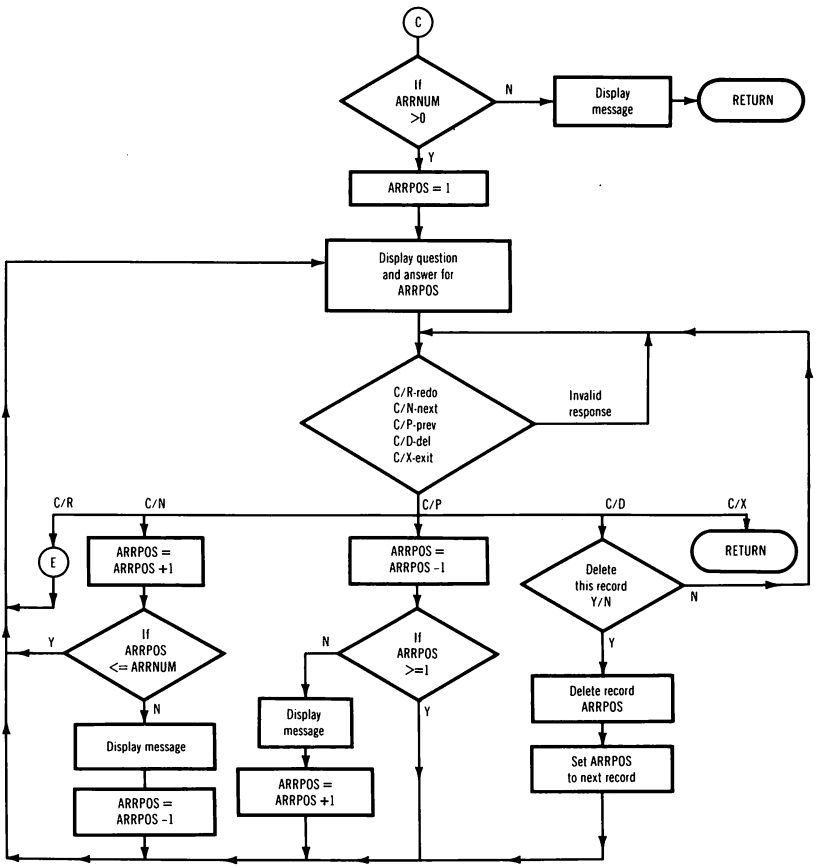
Flowchart 1 cont. — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base

(foldout)

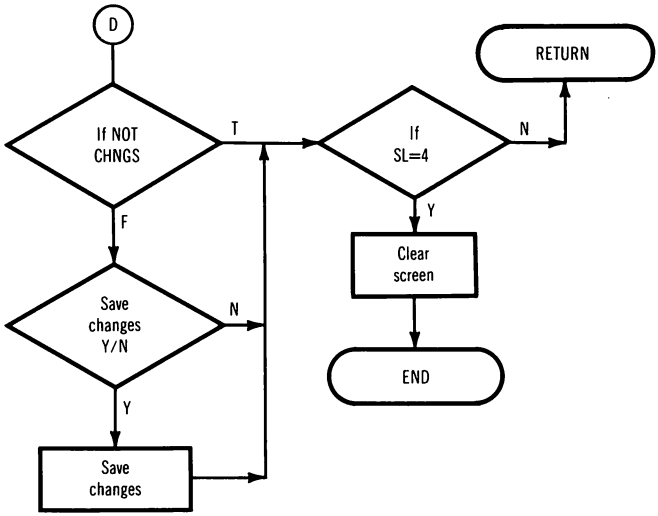
Flowchart 1 cont. — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base

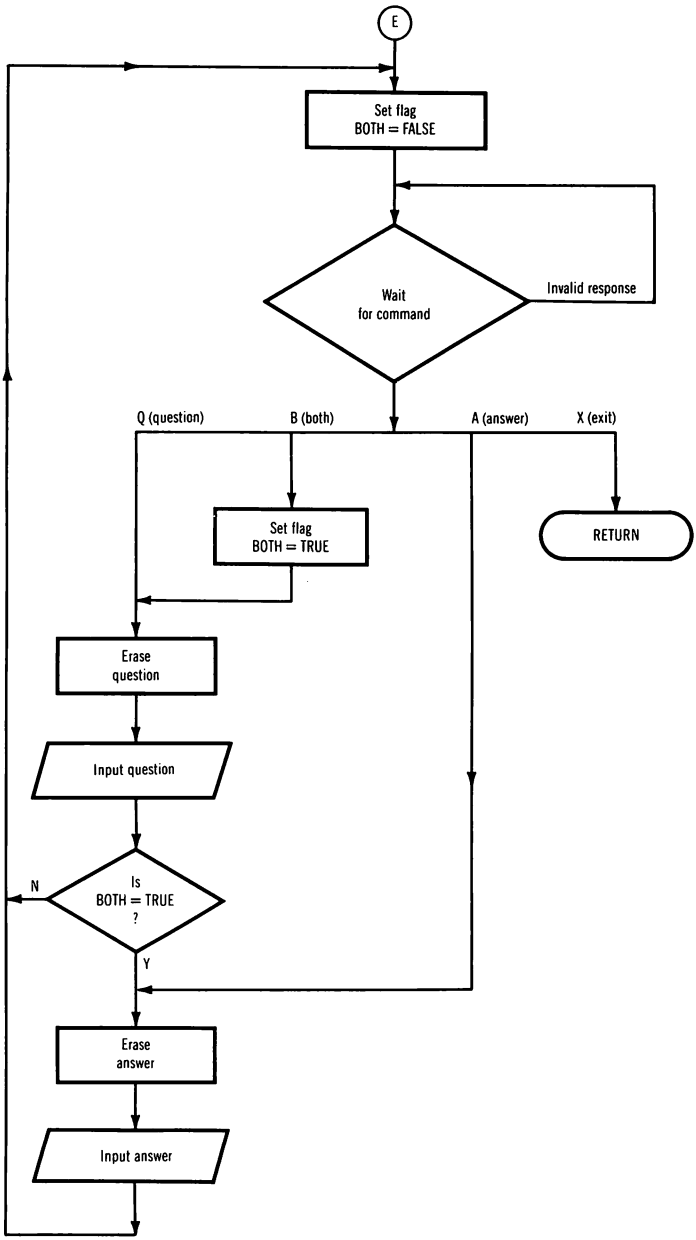
(foldout)

Flowchart 1 cont. — Cassette Data Base



Flowchart 1 cont. — Cassette Data Base (foldout)

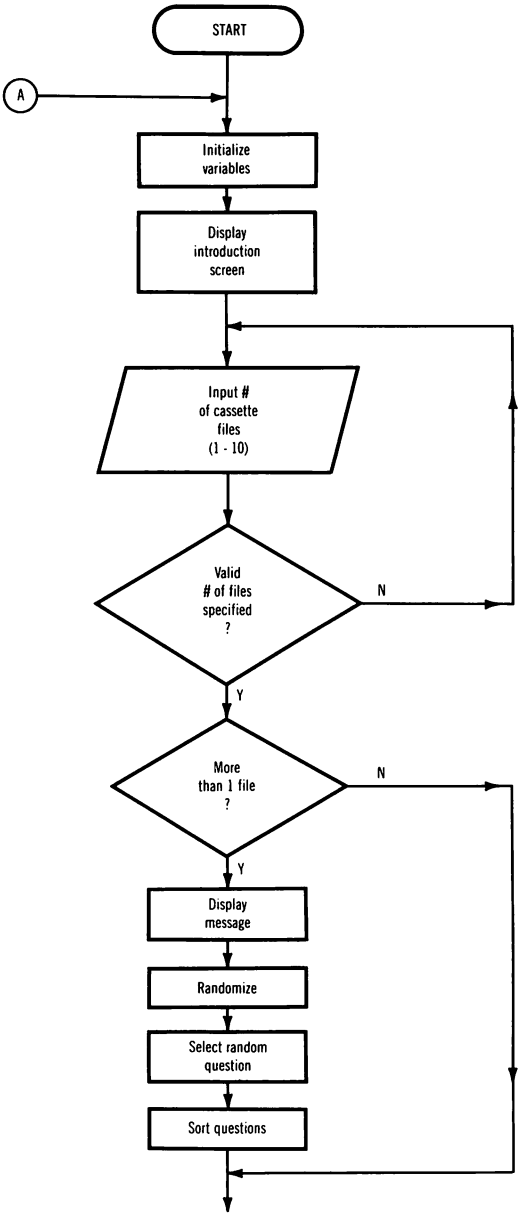
Flowchart 1 cont. — Cassette Data Base



Flowchart 2 — Cassette Game

(foldout)

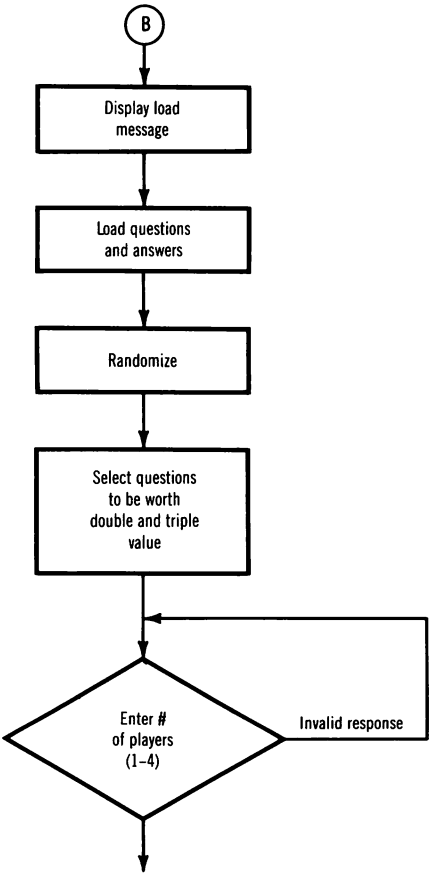
Flowchart 2 — Cassette Game



Flowchart 2 cont. — Cassette Game

(foldout)

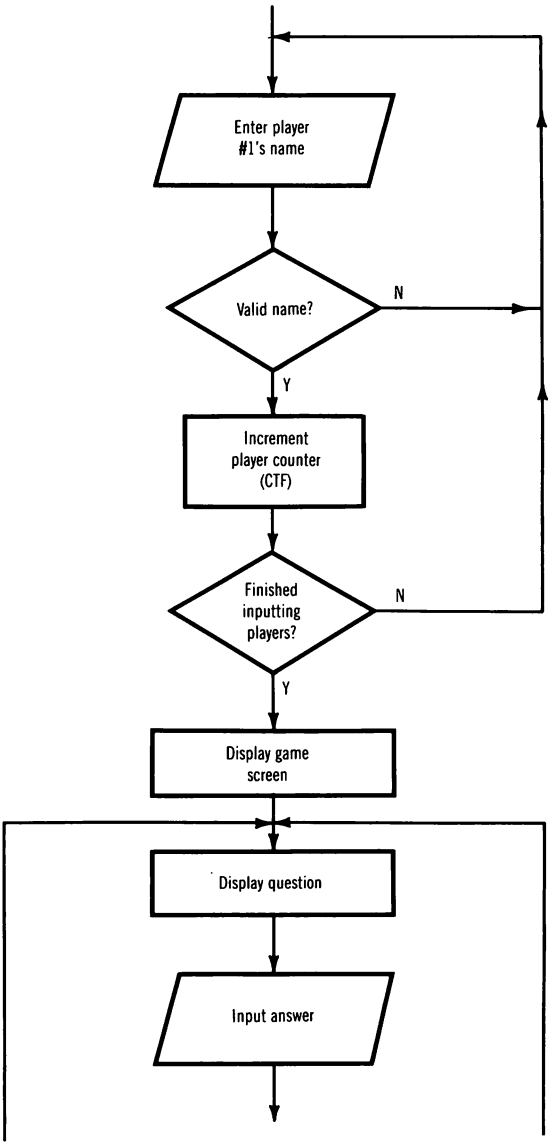
Flowchart 2 cont. — Cassette Game



Flowchart 2 cont. — Cassette Game

(foldout)

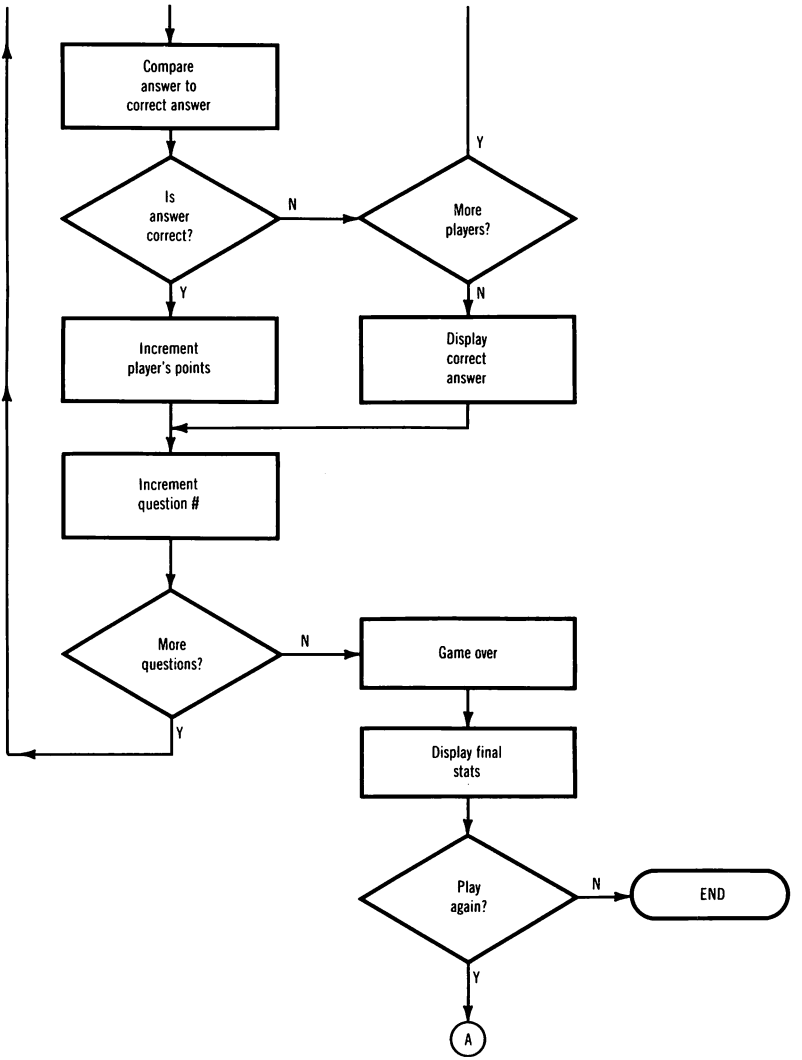
Flowchart 2 cont. — Cassette Game



Flowchart 2 cont. — Cassette Game

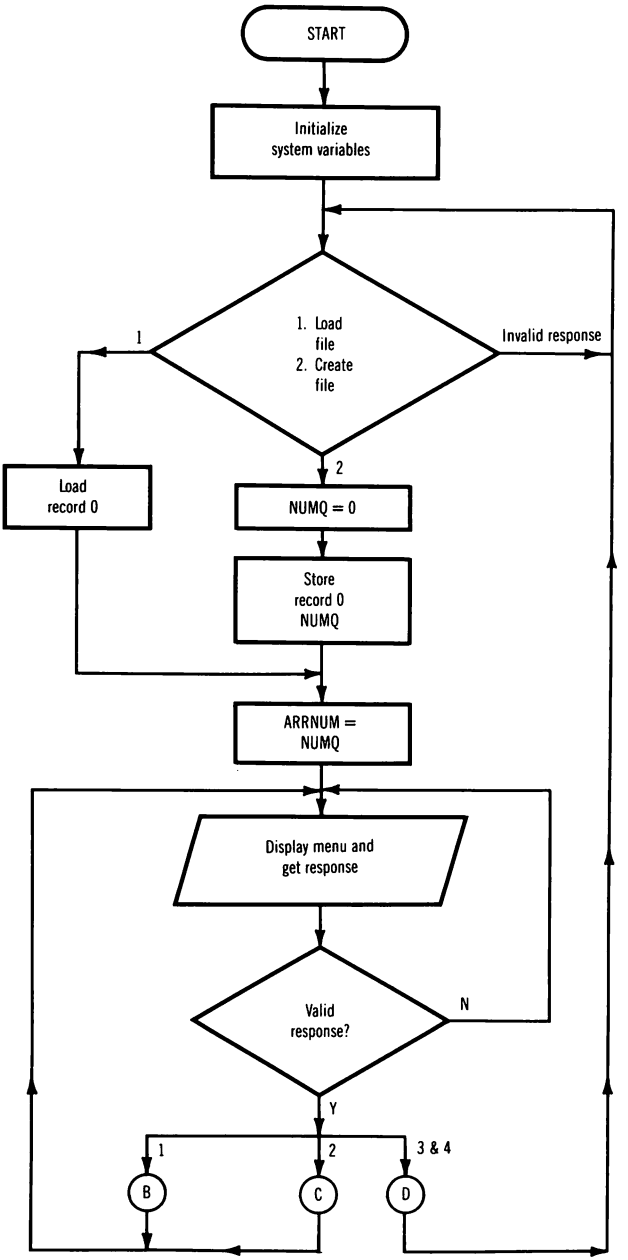
(foldout)

Flowchart 2 cont. — Cassette Game



Flowchart 3 — Diskette Data Base (foldout)

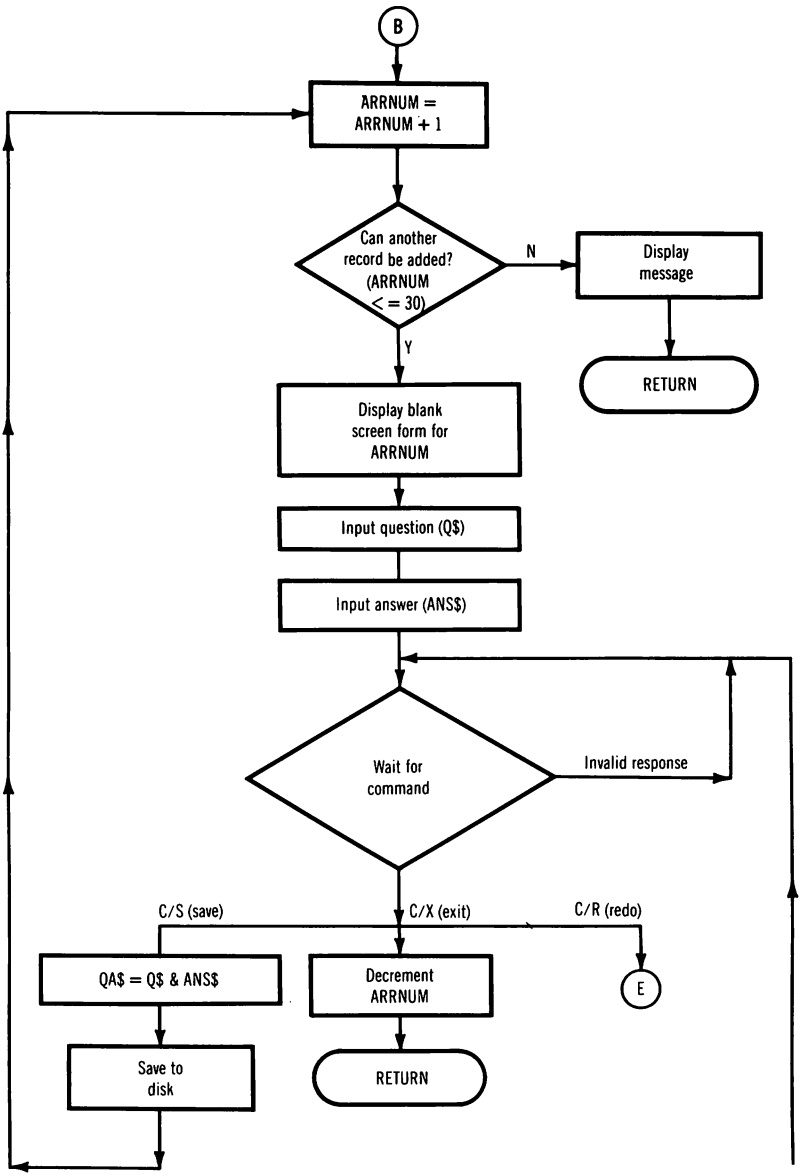
Flowchart 3 — Diskette Data Base



Flowchart 3 cont. — Diskette Data Base

(foldout)

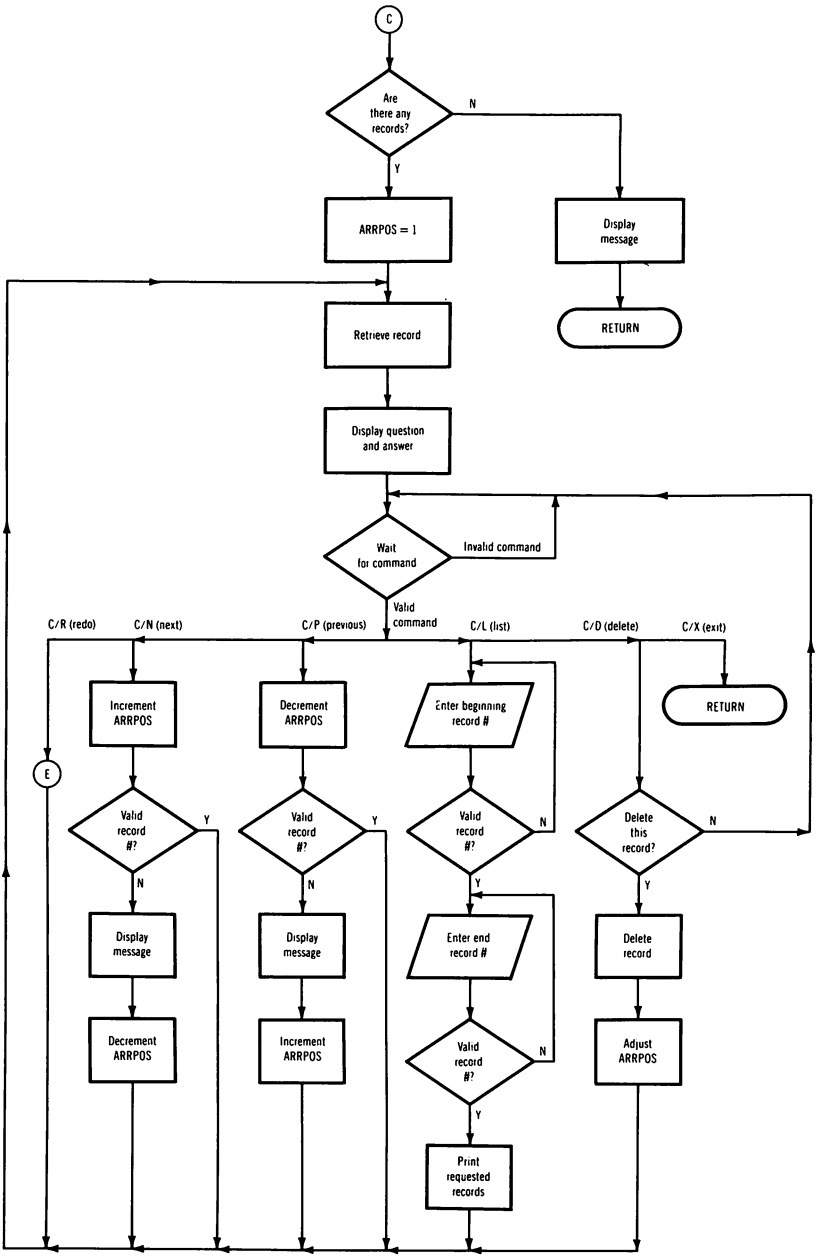
Flowchart 3 cont. — Diskette Data Base



Flowchart 3 cont. — Diskette Data Base

(foldout)

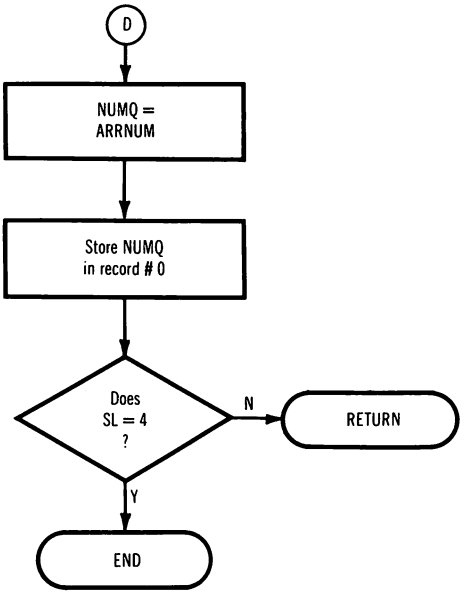
Flowchart 3 cont. — Diskette Data Base



Flowchart 3 cont. — Diskette Data Base

(foldout)

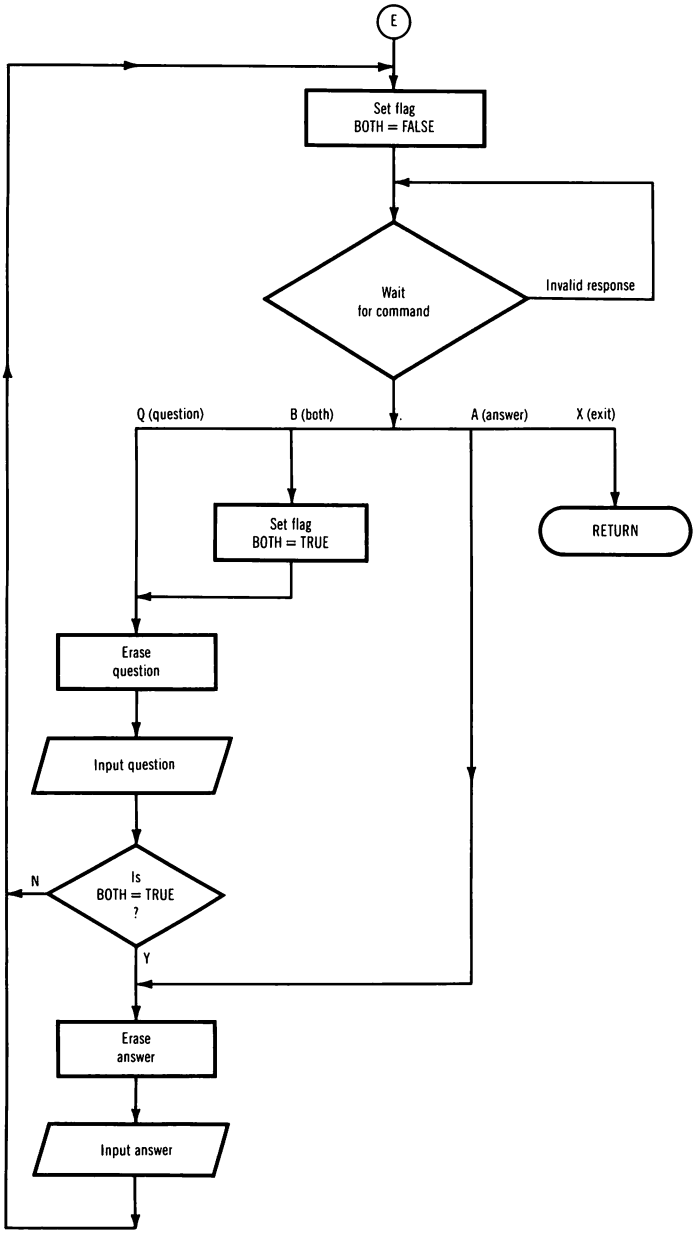
Flowchart 3 cont. — Diskette Data Base



Flowchart 3 cont. — Diskette Data Base

(foldout)

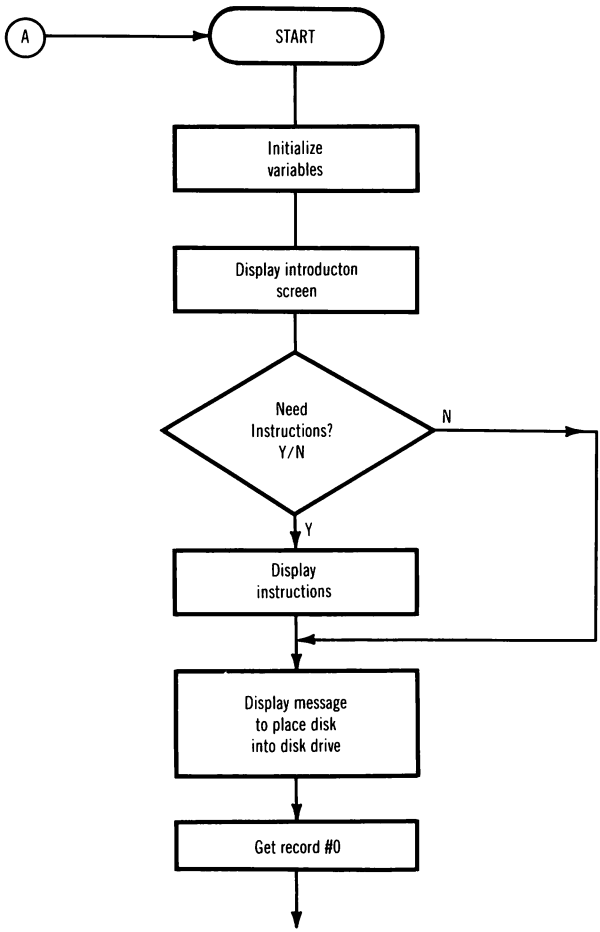
Flowchart 3 cont. — Diskette Data Base



Flowchart 4 — Diskette Game

(foldout)

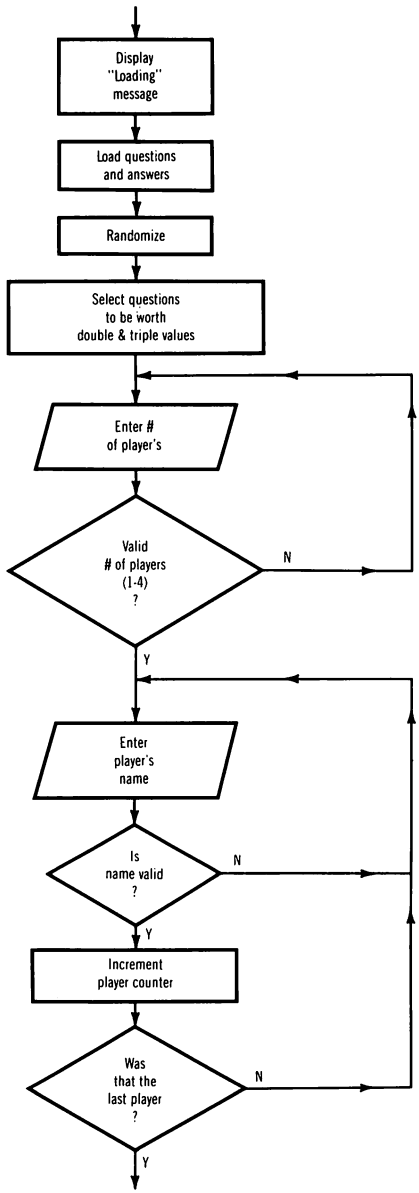
Flowchart 4 — Diskette Game



Flowchart 4 cont. — Diskette Game

(foldout)

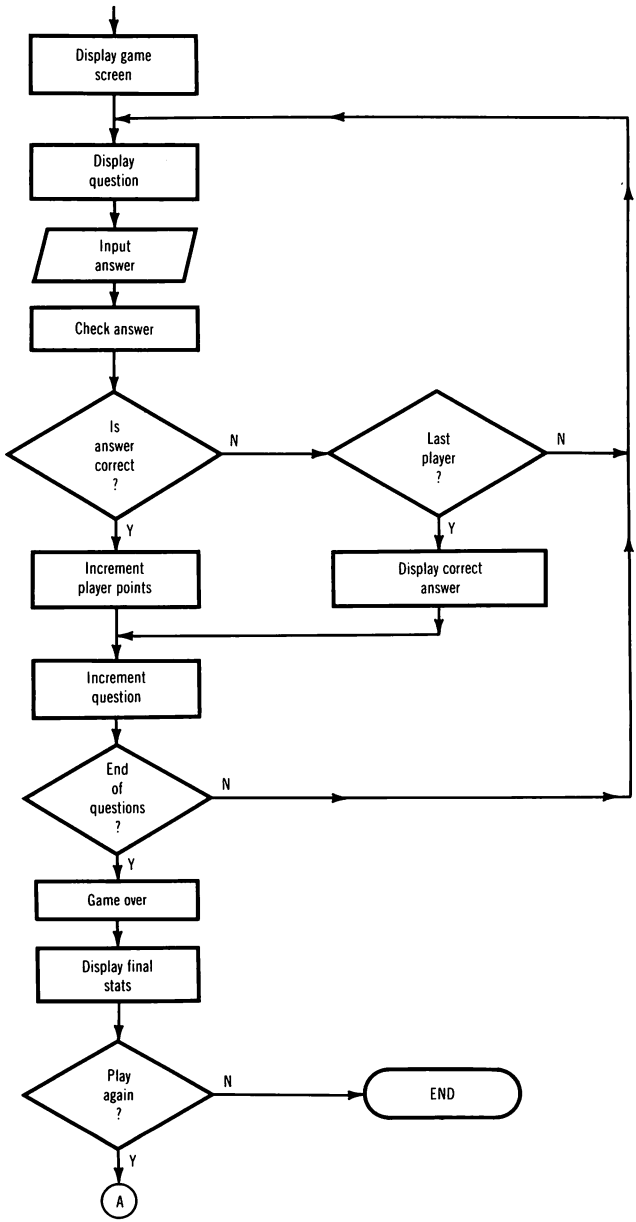
Flowchart 4 cont. — Diskette Game



Flowchart 4 cont. — Diskette Game

(foldout)

Flowchart 4 cont. — Diskette Game



SAMS

TI-99/4A™ Trivia Data Base

- Includes cassette and disk versions of two complete programs, a data base and a trivia game
- Introduces TI-99/4A owners to data bases and their applications by enabling users to understand how a simplified data base is created and why it works
- Discusses user-friendly data entry, error checking and program continuity
- Shows how to design individual file records
- Compares advantages and disadvantages of both cassette and disk files
- Provides tips on making programs run faster and more efficiently
- Offers education and entertainment for TI-99/4A users of all ages

Machine Requirements:
- **TI-99/4A Personal Computer**
- **TI Extended BASIC**
- **Cassette Player**

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

$8.95/22395 ISBN: 0-672-22395-3